AD-A252 404

REPORT DOCUMENTATION PAGE

Form Approved OPM No. 0704-0188

Public repc... response, including the time for reviewing instructions, searching existing data sources gathering and maintaining the data needed, ar... estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquart... Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Information and Regulatory Affairs, Office of Manageme...

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE January 1992	3. REPORT TYPE AND DATES COVERED Technical

4. TITLE AND SUBTITLE
Development Statistics for the UH-1 Ada Feasibility Study

5. FUNDING NUMBERS

6. AUTHOR(S)
IIT Research Institute
4600 Forbes Blvd.
Lanham, MD 20706

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
IIT Research Institute
4600 Forbes Blvd.
Lanham, MD 20706

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Ada Joint Program Office
OSD(A)ODDDRE(S&T)/AFF
RM 3E118/Pentagon
Washington, DC 20301-3081

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
N/A

JUL 0 2 1992

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

this document describes data collection and analysis techniques that were applied to an Ada software redevelopment of the UH-1 Flight Simulate or (FS). It presents results for the following subject areas:

- o The Application of function point analysis to estimate trainer size;
- o the application of Ada COCOMO, SoftCost Ada, and SASET models to estimate schedule and efforts;
- o the application of AdaMAT/D to evaluate trainer quality.

The report also describes results of a project profile study to characterize aspects of the development environment.

14. SUBJECT TERMS
Ada cost models Ada, software redevelopment, function point

15. NUMBER OF PAGES
78

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFED	UNCLASSIFIED	UNCLASSIFIED

NSN 7540-01-280-550

Standard Form 298, (Rev. 2-89)
Prescribed by ANSI Std. 239-128

DEVELOPMENT STATISTICS FOR THE

UH-1 FS ADA FEASIBILITY STUDY

January 1992

Prepared for:

PM TRADE
ATTN: AMCPM-TND-ED
12350 Research Parkway
Orlando, FL 32826-3276

Prepared by:

IIT Research Institute
4600 Forbes Boulevard
Lanham, MD 20706-4324

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF TERMS AND ABBREVIATIONS

ACS	Automated Courseware System
ADADL	Ada-based Documentation and Design Language
ADAMAT	Ada Measurement and Analysis Tool
CCC	Configuration Change and Control
CDR	Critical Design Review
COCOMO	Constructive Cost Model
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
DCE	Digital Conversion Equipment
DCL	Digital Command Language
DOCGEN	Documentation Generator
DR	Discrepancy Report
EDSI	Equivalent Delivered Source Instructions
FPA	Function Point Analysis
FQR	Final Qualification Review
FS	Flight Simulator
GAT	Generic Aircrew Trainer
IOS	Instructor Operator Station
IR&D	Internal Research and Development
KSLOC	Thousands of Source Lines of Code
NTSC	Naval Training Systems Center
MB	MegaBytes
NSIA CWG	National Security Industrial Association Computer Working Group
PDL	Program Design Language
PDR	Preliminary Design Review
PM	Person Month
PM TRADE	Program Manager for Training Devices
RFP	Request for Proposal
SASET	Software Architecture Sizing and Estimating Tool
SEL	Software Engineering Laboratory
SLOC	Source Lines of Code
SRR	System Requirements Review
SSR	Software Specification Review
STR	Software Trouble Report
TESTGEN	Test Generator
TRR	Test Readiness Review

(This Page is Intentionally Left Blank)

1.0 INTRODUCTION

1.1 BACKGROUND

The Army Material Command's Program Manager for Training Devices (PM TRADE) performs the principal role of acquiring training devices and training aides for the soldier to enhance operational proficiency and primary skills. The cost effectiveness of current development practices and a comprehensive methodology to improve the quality of the software in these automated systems have not been assessed or developed. As part of a tri-service initiative among PM TRADE, Naval Training Systems Center (NTSC), and United States Air Force Aeronautical Systems Division Deputy for Simulators, PM TRADE is performing research into the use of the Ada programming language to evaluate its impact in developing flight simulators.

To determine the feasibility of using new technologies for trainers, a baseline must be established against which the technology effects can be measured. Specific aspects of a software project are quantified to allow an organization to understand its development characteristics. A baseline is established as data are collected and projects are measured. Meaningful analyses of the data result in an improvement in an organization's understanding of the software development process within its environment and provides insight into parameters of interest such as productivity, maintainability, and cost. Subsequently, improvement in software development can be effected via the planned application and evaluation of new development technologies.

In the acquisition of a new system, especially where software is a sizable portion of it, a major problem of the developing organization is how to identify which software qualities are important, and then how to specify them in the form of requirements. As the system evolves during development, the need arises to determine how well those requirements are being satisfied. Each software system is unique in its level of software quality requirements. There are basic system characteristics which affect the quality requirements, and each system must be analyzed individually for its fundamental characteristics.

1.2 OBJECTIVE

Our basic goal was to implement an Ada Data Collection and Analysis Program and coordinate the program with the development contractor to fulfill the goals of defining productivity, cost, and quality metrics to support future acquisitions. This paper describes our approach to data collection and analysis, and how techniques were applied to one particular trainer development project, the UH-1 Flight Simulator. The results of the study should be a greater understanding of the software development process, product improved simulators, and associated specifications and contracts.

1.3 APPROACH

We chose an approach that combines a practical, proven methodology for measuring software quality with experiments that are designed to measure differences in Ada programming practices that impact productivity and software quality. The proven methodology is one that was designed and developed at the NASA/Goddard Space Flight Center Software Engineering Laboratory (SEL). The SEL was founded in 1976 to carry out studies and measurements related to evolving technologies in the flight dynamics area. In 1985, the SEL initiated an effort to study the characteristics, applications, and the impacts of Ada. The SEL has subsequently collected detailed development data from a total of eight Ada projects. The goals of the SEL are to understand the software development process; measure the effects of various methodologies, tools, and models on this process; and then to identify and apply successful development practices. So that the expense of data collection does not get out of hand, their major emphasis is to define measurement goals and let the goals drive the data that are being collected [1,2].

Data collection and analysis for the UH-1 FS Ada Feasibility Study focused on five measurement objectives:

1. Provide a profile to characterize aspects of the development environment.

2. Quantify some of the effects of Ada on measures of significant importance such as productivity, reliability, reuse, and maintainability.

3. Determine how the trainer development that is the target of this study compares to trainer developments in non-Ada languages.

4. Determine how to best estimate the cost of trainer software development in Ada.

5. Evaluate the feasibility of using function points to estimate trainer size.

The data required to support UH-1 measurement objectives were collected by using the following five methods: 1) completion of a data collection form by the developer, 2) observation of development, 3) code analysis, 4) interviews with the developer, and 5) review of project software documentation. The data collection form was maintained throughout the project and updated at major milestone reviews. The completed form is contained in Appendix A. Instructions for completing the form are in Appendix B. The form contains information that is not only used to support the project profile study, but it also supports application of three cost estimation models.

Software cost estimation models were applied at three different intervals throughout the project in order to update schedule and effort projections. When model results were reviewed at PDR, there was much discussion on the estimated size of the program which was believed to be too high. The fifth measurement objective, evaluating function points as a means to estimate trainer size, was an outgrowth of these discussions.

An enhanced Intermetrics Statement Profiler is one of the automated tool used for code analysis. By passing source code through the tool's parser, it will count the usage of various Ada statement types and special Ada features. The tool currently operates on a VAX system under VMS. The tool makes use of a parser that was previously developed by Intermetrics, Inc. for their 'Statement Profiler' tool. The 'Statement Profiler' is available from the Ada Software Repository. The Intermetrics tool was enhanced to include additional output counts and user interfaces were changed to make them easier to use. In some cases, some constructs were not considered in the parser. In these instances, a character string that uniquely identifies the construct is flagged by the tool and the appropriate counter is incremented.

1.4 REPORT ORGANIZATION

The following sections describe the studies and experiments that were performed, and data that was collected to support the analysis. Each major section addresses a separate measurement objective:

- Section 3.0 presents the results of a project profile study to characterize aspects of the development environment.

- Section 4.0 describes the application of function point analysis to estimate trainer size.

- Section 5.0 describes the application of Ada COCOMO, SoftCost Ada, and SASET models to estimate schedule and effort.

- Section 6.0 presents the results of the trainer quality evaluation that was based on an evaluation of software changes and errors and on the application of AdaMAT/D.

When available, results from other measurement programs are presented to provide a basis for comparison. We were unable to acquire trainer-specific data for a comparative project study. A brief overview of the UH-1 FS system is presented in Section 2.0. A summary of the lessons learned from the measurement process are presented in Section 6.0

2.0 SYSTEM OVERVIEW

The UH-1 FS is a flight simulator for the UH-1H helicopter. It teaches instrument flight maneuvers and procedures as well as normal and emergency cockpit procedures to Army aviators. The UH-1 FS consists of four independently operated helicopter cockpits, a central two position instructor console, a digital computer system and some ancillary equipment. Each cockpit has its own five-degree-of-freedom motion system and a sound system.

The UH-1 Flight Simulators which were designed in the late 1960's had become difficult and expensive to maintain. The spare memory and spare CPU time had been depleted by software changes. The Army proposed a product improvement plan to swap-out the aging UH-1 FS computer system to improve its capability to be supported and to provide a means to split the 4-cockpit trainer into two 2-cockpit trainers, if needed.

There are two computer software configuration items (CSCI's); namely, a Real-Time CSCI which performs all simulation and processing functions of the UH-1 FS and a Non-Real-Time CSCI which contains all support, diagnostic, and courseware for the UH-1 FS. The latter encompasses the Ada Programming Support Environment (APSE), the Automated Courseware System (ACS), any database/support software and any commercial software tools.

The Automated Courseware System (ACS) software is the component providing the capability to develop and modify trainer courseware via the Automated Courseware workstation. The ACS provides for the formulation and editing of UH-1 FS mission scenarios consisting of navigational aides, initial operating conditions, and real-time maps. Courseware data entry tasks are performed on the workstation and are transferred to the real-time systems (at distributed sites) via a courseware floppy disk.

The development computer system was a 16 MB MicroVAX II operating under a VMS operating system. Tools which supported the development environment included a TeleSoft Ada compiler, ADADL, DOCGEN, TESTGEN, and CCC. A VAX/VMS hosted Ready Systems RTAda cross compiler was used to generate object code for an MC68030 target system. The target computer system was a network of loosely coupled MC68030 processors operating as one six node configuration (with a single node allocated to each of the four cockpits and a single node allocated to each half of the instructor operator station) or as two three-node configurations when the trainer is split. Each node performs the bulk of the real-time simulation for the cockpit or Instructor Operator Station (IOS) node locally so as to minimize the amount of data passed between the nodes in real time.

The ACS system console terminal incorporates a single-board Motorola 68030 series microcomputer along with Motorola memory and interface boards.

3.0 CHARACTERIZING THE DEVELOPMENT ENVIRONMENT AND ITS PRODUCTS

Analyses of detailed profile information that characterize aspects of the development environment and the products of that environment are useful to better understand the software development process for an application domain. Profile studies are not designed to evaluate whether the characteristics are right or wrong but to report on the method of software development [1]. The following sections describe characteristics of the UH-1 FS program relating to source code attributes, phase distribution of effort and schedule, and productivity.

3.1 COMPARISON OF SOURCE CODE SIZE

Table 3-1 contains a comparison of source code size when different counting conventions are used to provide sizing information. Line counts were obtained by applying an enhanced version of the Intermetrics Statement Profiler on baseline version 15 of the source code which was the version of software shipped to the first trainer installation site, one month prior to Government Final Inspection. A description of counting conventions for physical lines, terminal semicolons, body semicolons, and essential semicolons is provided in Appendix C. The body semicolon count is 60 percent larger than the terminal semicolon count. The large

TABLE 3-1

UH-1 FS SOURCE LINES OF CODE SIZE

Deliverables Program	Language	Physical Lines	Terminal Semicolons	Body Semicolons	Essential Semicolons	Software Type
IOS CSC	Ada	49,809	11,671	15,591	3,546	Application
Trainee Station CSC	Ada	49,802	12,489	18,722	7,511	Application
ACS CSC	Ada	24,774	6,523	9,905	1,870	Application
	Ada	6,242	1,773	1,859	1,529	Support
Common to ACS, IOS and Trainee Station	Ada	46,305	6,159	25,639	2,709	Application
	Ada	387	97	97	67	Support
	DCL	613	584	584	N/A	
	Assembly	8,919	6,545	6,545	N/A	
DCE Diagnostics	Ada	14,962	7,334	7,481	910	Support
Target Computer Diagnostics	Ada	7,081	1,061	3,561	113	Support
Daily Readiness	Ada	5,934	1,187	1,253	646	Support
TOTAL		**214,828**	**55,423**	**91,237**	**18,901**	

size differential between body semicolons, which include the number of carriage returns in each package specification, and terminal semicolons is attributed to the general packaging framework used on the UH-1 program which resulted in about three package specifications for each body.

The completed project questionnaire in Appendix A shows the categorization of code that are new, reused/modified, and reused/unmodified for terminal and body semicolon counts.

3.2 SOURCE CODE ATTRIBUTES

Table 3-2 contains a summary of source code attributes. Statement counts were obtained by applying an enhanced version of the Intermetrics Statement Profiler. Definitions for Ada statement types are provided in Appendix C.

TABLE 3-2

SOURCE CODE ATTRIBUTES

Program Units	
Number of Objects	3,908
Number of Packages	322
Number of Tasks	33
Number of Program Units	1,517
Number of Blocks	2
Statement Types	
Number of Logicals	13,846
Number of Data Manipulations	14,461
Number of Ada Tasking	63
Number of Data Typing	2,138
Number of Mathematical	8,733
Number of Declarations	4,041
Ada Features Data	
Number of Exit Statements	147
Number of Use Clauses	1,269
Number of Exceptions	351
Number of Generics	16

3.3 USE OF ADA FEATURES

In an effort to achieve some measurement of the use of the features available in the Ada language, the SEL identified six Ada features to monitor: generic packages, type declarations, packages, tasks, compilable PDL, and exception handling [2]. The SEL then examined the code to see how little or how much of these features were used. The purposes of this analysis were, first, to determine to what degree features of Ada were used by the Ada project, and second, to determine whether the use of Ada features "matured" as an environment gained experience with the language. SEL data on the use of Ada features were obtained using the Ada Static Source Code Analyzer Program developed at the University of Maryland. Analysis of the use of compilable PDL and exception handling did not show any trends, however, trends were observed in the use of other features. Figure 3-1 show SEL trends in the use of Ada features over a span of seven years, beginning with their first Ada project in 1985. A total of eight Ada projects are included in the trend analysis [3]. Ada features data for the UH-1 FS are included in the trend analysis for comparison.

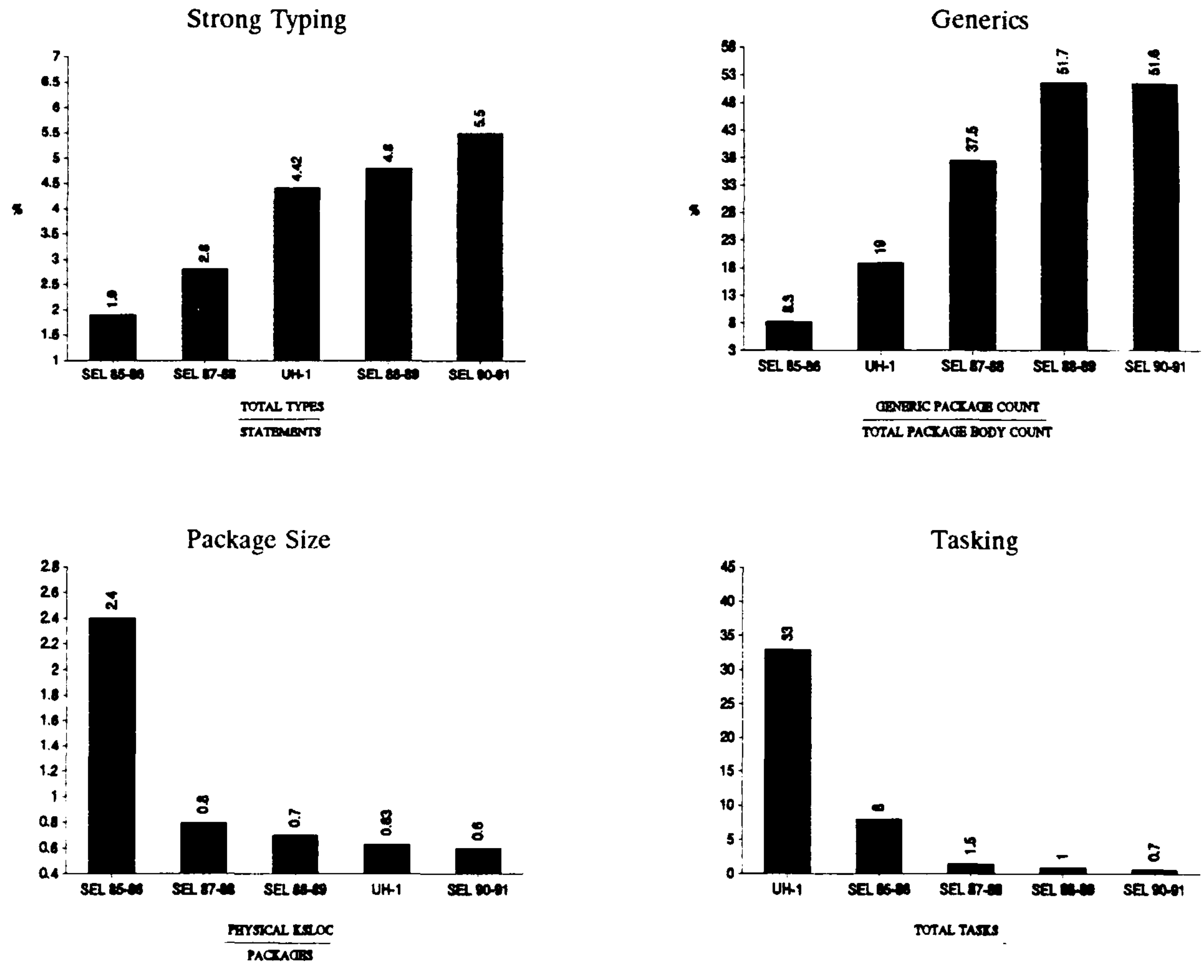


Figure 3-1. Use of Ada Features - Comparison to SEL Data.

The use of strong typing in these software systems was measured by the number of type and subtype declarations divided by the number of Ada statement (terminal semicolon count) multiplied by 100 to obtain a percentage. It is generally believed that the strong typing of Ada will prevent some types of interface errors. The measure provides a method of observing trends in the use of Ada type declarations. In the flight dynamics environment, the amount of typing has increased over time. This has been attributed to the developers becoming more comfortable with the strong typing features of Ada and using its capabilities to a fuller extent [2,3]. The proportion of type declarations to Ada statements on the UH-1 was 4.42 percent.

The generic package is a tool in the Ada language that contributes to software reusability. The SEL has placed a strong emphasis on the development of reusable components and has seen an increase in the use of generic packages from the first to the current Ada project. Additionally, the SEL trends reflect an increased understanding of how to use generic Ada packages effectively in a flight dynamics environment. It is currently perceived that the proportion of generic packages to total package body count will level off at about 50 % on future projects as the SEL reaches the limit at which existing program units can be generalized [2,3]. There were a total of 16 generic packages developed on the UH-1 program representing 19 percent of the total package body count. The use of generics on the UH-1 project, which was the development team's first Ada project, was favorably comparable to the trends documented by the SEL.

The average size of packages was measured by dividing the number of physical Ada lines of code by the number of packages. The SEL trends show an average size of the packages for the first Ada projects are much larger than the average size for subsequent Ada projects. The variation is due to a difference in the structuring method between the first Ada project and all subsequent Ada projects. The first Ada project was designed using a heavily nested structure with one package at the root of each subsystem and where package specifications were nested with package bodies. Subsequent projects were designed utilizing the view of subsystems described by Grady Booch as an abstract design entity whose interface is defined by a number of separately compilable packages [2,3]. UH-1 FS design methodology is consistent with the latter. The average size per package on the UH-1 was 630 physical lines.

A comparison of tasking between applications in the flight dynamics environment and the trainer environment indicate that the tasking feature of Ada is highly application dependent. The use of this Ada feature at the SEL has declined with each successive Ada project as personnel have learned to use tasking only in those situations that are appropriate [2,3]. A total of 33 Ada tasks were implemented on the UH-1 FS.

3.4 PHASE DISTRIBUTION OF EFFORT AND SCHEDULE

Phase distribution of effort entails the allocation of staff throughout the requirements, design, implementation and testing phases of the development cycle. Using milestone dates to denote the end of one phase and the beginning of the next, the UH-1 FS project showed nearly 40 percent of the total effort was expended prior to CDR and approximately 60 percent was expended after CDR. The phase distribution of effort was contrary to other published data [4] that indicates a shift of effort from the integration and test phases to design phases. Tables 3-3 and 3-4 illustrates the traditional allocation of time and effort to life-cycle phases for two Ada-specific models: Ada COCOMO and SoftCost Ada. The 41:59 distribution of effort before and after CDR for the UH-1 compares to a 50:50 distribution of effort for SoftCost Ada and a 52:48 distribution of effort for Ada COCOMO. The differences are attributed to four factors:

1. The UH-1 is a redevelopment of an existing system whereas effort distributions for the models are based on new development efforts. It is reasonable to expect less time spent on defining requirements on a redevelopment effort as compared to a new development. To illustrate, the system requirements for the UH-1 were to "replicate current UH-1 FS functions and performance unless stated otherwise". This resulted in less time spent communicating requirements between the developer and the sponsor.

TABLE 3-3

PHASE DISTRIBUTION OF EFFORT (%)

	SRR - SSR	SSR - PDR	PDR - CDR	CDR - TRR	TRR - FQR
ADA COCOMO	N/A	23	29	22	26
SOFTCOST ADA	50			15	35
UH-1	16		25	59	

TABLE 3-4

PHASE DISTRIBUTION OF SCHEDULE (%)

	SRR - SSR	SSR - PDR	PDR - CDR	CDR - TRR	TRR - FQR
ADA COCOMO	N/A	39	25	15	21
SOFTCOST ADA	50			15	35
UH-1	7	11	23	45	14

2. Phase distribution of effort for the UH-1 includes systems integration and testing to obtain a fully functioning hardware-software system whereas effort distribution for the models does not cover implementation.

3. The milestone dates were specified in the Request for Proposal (RFP). Further, dates for PDR and CDR were specified as payment milestones for the developer. In an effort to meet payment, milestones were scheduled earlier than what may have otherwise been considered optimum. For example, the developer did not have compilable package specs by PDR which is one of the highlights of the Ada Process Model, which is the basis of the Ada COCOMO model [4].

4. The developer used a structural model design methodology. A structural model is a domain specific software architecture. Expectations are that structural model designs are transitional and reusable for similar types of applications, i.e., flight simulators [5]. The developer utilized the concept of the structural model in a Generic Aircrew Trainer (IR&D) project. During the development of the Ada code for the Generic Aircrew Trainer (GAT), various methodology problems were uncovered. Work on the GAT enabled the developer to iron out specific details of the structural model to be applied to the UH-1 that would have otherwise been charged to the project.

A comparison of the phase distribution of effort to phase distribution of schedule indicates a consistent staffing across the project with 41 percent of the time spent prior to CDR and 59 percent of the time spent after CDR.

3.5 PRODUCTIVITY

Because so many definitions exist for software size measures in Ada, it is important that any productivity value be qualified by the basis for the measure. We measured productivity on the UH-1 program using two definitions: 1) terminal semicolons, and 2) body semicolons. We chose the first because sources show it to be a more widely used definition. However, in the case of the UH-1 FS, productivity measurement based on terminal semicolons penalize the developer because of the packaging structure which was used.

The general packaging framework was that each object in the system consisted of one message (i.e. package) specification and one body with associated message specs. The message specification defined the status of the object at any given time and contained only that information which was exported to other portions of the system. Each object could have as many as four additional message specifications: DCE input and output specifications that interface with Digital Conversion Equipment handlers/drivers, action request specification that interface with malfunction control, and a test points specification used to access intermediate test point variables to allow strip chart recording of variables during flying qualities tests. Since terminal semicolons are not used in package specifications, productivity measures that are based on terminal semicolon count penalize the developer that uses package specifications as the primary means of communication. The body semicolon count for the UH-1 program was 60 percent larger than the terminal semicolon count (reference Table 3-1).

With reused software factored in, the productivity for delivered Ada code on the UH-1 FS redevelopment project is shown in Table 3-5. The productivity was high considering that this was the first Ada project for the development team. (Only one of the lead designers had worked previously on the GAT.) Factors that are believed to have influenced productivity are that this effort was a redevelopment as opposed to a new development and that the structural model design methodology was partially reused from the GAT. Section 3.4 discusses these factors in more detail.

TABLE 3-5

UH-1 PRODUCTIVITY

Productivity = 200 EDSI / PM	**Productivity = 316 EDSI / PM**
EDSI counted in terminal semicolons	EDSI counted in body semicolons
Developed code = new code + 16 percent reused/modified code	
Hours per person month = 152	
Period extends from SRR to FQR (i.e., installation at first trainer site) and includes implementation	

4.0 ESTIMATING TRAINER SIZE USING FUNCTION POINT ANALYSIS

At a project Lessons Learned briefing held subsequent to PDR, the results of the application of SoftCost-Ada were presented to project sponsors and the developer. There were three issues raised with regard to the validity of the estimates. Two addressed the impact of the structural model on productivity and the existence of analogous data in the SoftCost-Ada database. The third was critical of the size of the project, which the developer believed to be too high. It was decided to apply additional cost models when schedule and effort projections were updated at the next milestone and to use function point analysis to estimate trainer size.

Table 4-1 provides a history of the size estimates that were made at project milestones beginning with the projection made by the developer in the proposal. Although the UH-1 FS Ada Feasibility Study was a redevelopment of an existing system, it was impossible to derive an estimate from the existing system because of an inability to determine what source listings matched the executable software. The software was written in assembly and, over the years, many modifications were patched onto the system. The estimate at PDR was provided by PM TRADE based on similar FORTRAN trainer applications. The estimate at CDR was based on function point analysis and did not include support software. The following points summarize a few observations relative to the sizing history:

- There is a tendency to underestimate support software. In all cases estimates for the Non-Real-Time CSCI, which included all trainer support software, were low by a factor of 40% or more.
- Although the CDR estimate for total KSLOCS was very close to the actual terminal semicolon count, the proposal estimate was the best estimate for individual CSCIs.
- The proposal and PDR estimates support the notion that we tend to estimate in terminal semicolons as opposed to body semicolons. In all cases, estimates were comparable to the actual size based on the terminal semicolon counting convention.

TABLE 4-1

UH-1 SIZING HISTORY

	Real-Time CSCI	Non-Real-Time CSCI	Total KSLOC
Proposal	31.1	15.6	46.7
PDR Estimate	24.6	7.1	31.7
CDR Estimate (Function Points)	44.7	7.3 +	52.0 +
Actual (Terminal Semicolons)	30.3	25.1	55.4
Actual (Body Semicolons)	59.9	31.2	91.2

4.1 PROCEDURE FOR ESTIMATING SIZE USING FUNCTION POINTS

Function point analysis (FPA) measures an application by quantifying the information processing function associated with five data types: external inputs, external outputs, external inquiries, logical internal files, and interfaces. Obtaining the trainer size estimate was accomplished in three steps:

1. Compute the unadjusted function point measure by classifying and counting the five data types

2. Adjust for processing complexity (+/- 35%)

3. Apply the language expansion factor.

The function point total is a unitless measure of the functionality of the software, independent of lines of code or implementation language. Several sources have observed a relationship between function point measures and the SLOC estimate for the implementation language [6]. For example, two programs of identical function are implemented in two different languages, FORTRAN and Ada. The function point measure for each program is the same at 100. Using a language expansion factor of 71 for Ada and a factor of 105 for FORTRAN, the same program implemented in Ada takes 7,100 SLOC and 10,500 lines in FORTRAN.

Initially, language expansion factors exemplify typical values for an organization based on the developer's particular dataset. Variations in programming skill, programming style and function point counting conventions will result in different language expansion factors for the same language. These factors may require modification after the user has applied the model successively and has evaluated the estimated versus actual size.

Two function point estimates were derived for the UH-1 FS: one for the real-time trainer application software and one for the Automated Courseware System (ACS). Support software that was not included in the function point estimate is listed in Table 4-2. The size of each support software component is provided in terminal semicolons. Support software consisted of anything having to do with setting up the training environment. There was some difficulty in determining what software constituted support software. The definition that was adopted was anything having to do with setting up the environment was considered to be support software.

Appendices D and E describe the process to identify function point parameters for the Real-Time CSCI and ACS, respectively. The appendices illustrate conventions that were adopted for identifying and counting function point parameters. Interpreting the guidelines [9] to define and count function point parameters, and extending the guidelines to training devices was not a strait-forward process. The greatest difficulty was determining how instrument display devices, malfunctions, and various flight controls should be grouped and counted. Examples of the conventions that were adopted are as follows:

- Group switches that work in conjunction with one another and count them as one input. For example, the UHF Radio Set consists of six control: 1) function selector switch, 2) mode selector switch, 3) preset channel control, 4) ten megahertz control, 5) one megahertz control, and 6) five hundredths megahertz control. The UHF Radio Set Controls were grouped and counted as one input rather than counted as separate inputs.

- Group malfunctions according to the object that they affect rather than count each malfunction separately. Hence, 113 malfunctions were grouped into 17 malfunction groups, i.e., fuel system malfunctions, malfunctions affecting VHF navigation, instrument malfunctions, malfunctions affecting engine lubrication, etc.

- Group instrument display devices according to the type of information that is displayed rather than count each instrument display device as a separate output. For example, fuel quantity was counted as one output displayable on four separate indicators: 1) minutes of fuel remaining - digital readout, 2) fuel quantity indicator, 3) auxiliary fuel low caution light, 4) 20 minutes fuel remaining caution light.

TABLE 4-2

SUPPORT SOFTWARE NOT INCLUDED IN FUNCTION POINT ESTIMATE

Deliverable Program	Description	Size
Target Computer Diagnostics	Tests main simulation computer equipment or ACS computer equipment. Checks computer configuration and its options, all memory units, peripheral units, and input/output units.	1,061
DCE Diagnostics	Performs functional checkouts of all trainer interface hardware controlled by computer system with test values characteristic of real-time operation.	7,334
PROM Related	Assembly software used for booting the system	6,545
Daily Readiness	Checks out all trainer equipment to see if trainer is ready for daily operation. Determines if all discreet and analog inputs and outputs are operational.	1,187
Disk Partitioning	Partitions the disk for real-time and non-real time loads.	97
Courseware Loader	Provides for transfer of courseware from the ACS to the Real-Time CSCI.	1,773
Floppy Disk Initialization	Formats the floppy disk on which courseware files generated by the ACS are loaded.	
Command Procedures	Miscellaneous DCL command procedures.	584
Total Statements		18,581

4.2 RESULTS

An Ada language expansion factor of 71 was used estimate size from the function point measure. Table 4-3 show a comparison of the function point estimate to the actual size. Actual size refers to application code which was counted using the terminal semicolons counting convention. The function point estimated sizes were high for both the ACS and the Real-Time CSCI. The estimate for the ACS only had a relative error of 11 percent as compared to the 32 percent relative error of the Real-Time CSCI. This is probably due to the fact that function point analysis has historically targeted the ACS type of application, i.e., management information systems. A comparison of the actual size of the Real-Time CSCI to the estimated size shows that an Ada language expansion factor of 48 (i.e., 1 function point = 48 SLOC) would have been appropriate for this application.

TABLE 4-3

COMPARISON OF FUNCTION POINT ESTIMATE TO ACTUAL SIZE

Deliverables Program	Actual Size	Function Point Estimated Size	Relative Error
ACS CSC	6,523	7,304	+ 11%
Real-Time CSCI	30,319	44,731	+ 32%

5.0 ESTIMATING TRAINER COSTS

One of the measurement objectives of the UH-1 FS Ada Feasibility Study was to determine how best to estimate development costs and schedule. There were several factors that would influence the study which are discussed in Section 3.4, namely,

- The UH-1 is a redevelopment of an existing system.
- The milestone dates were specified in the RFP.
- The developer used a structural model design methodology which was developed on a previous IR&D project and applied to the UH-1.

Although it was not known how much of an influence these factors would have on productivity and schedule, it was decided to utilize cost models as though the project were a new development.

Three models were applied at CDR, and FQR as follows:

- Ada COCOMO as implemented by COSTMODL (version 5.1)
- SoftCost Ada (version 2.1)
- SASET (Software Architecture Sizing and Estimating Tool - version 1.7)

The SoftCost Ada model was also applied at PDR. These particular models were chosen based on their availability to project personnel.

Model inputs were provided by the developer in the form of a project questionnaire which was maintained throughout the project and updated at major milestone reviews. The completed project questionnaire is provided in Appendix A and identifies the model(s) to which each question applies. The size data was obtained from baseline version 15 (referred to as the "Cold Start" tape) of the developer's software which was the version of software shipped to the first training site in Los Alimitos, California. This version of the trainer was fully tested with the exception of the motion system. The questionnaire provides lines of code counts using both terminal semicolons and body semicolons counting conventions. The terminal semicolon count was input to the SoftCost Ada and SASET models. The body semicolon count was used as input to the Ada COCOMO model.

Tables 5-1 and 5-2 provide a comparison of each model's schedule and effort projections to the actual project resources expended by the software developer. Table 5-1 shows costs for software development, excluding implementation. Since there was not a "clean" break between the time that software was completed and hardware/software integration began (activities were concurrent), the effort and 31 month schedule for software development is estimated. Table 5-2 includes the implementation phase, therefore, Ada COCOMO estimates do not apply.

In general, the model projections for effort were much higher than the actual effort expenditure reported by the developer. It is believed that the factors discussed previously (i.e., redevelopment versus new development, reused structural model design) had a significant impact on productivity. However, additional data points would be needed to validate this assumption.

SASET allows the user to run the model, optionally specifying the CDR date. The scheduling algorithms used by the Ada COCOMO and SoftCost Ada models, and SASET - when CDR was specified -

TABLE 5-1

COSTS FOR SOFTWARE DEVELOPMENT

	EFFORT (PM)	SCHEDULE (MONTHS)	FULL-TIME STAFF
ADA COCOMO	308.9	30.5	10
SASET	384	35	10
SASET (CDR SPECIFIED)	433	38	11
SOFTCOST ADA	410	33.3	12
UH-1	227 (estimated)	31	7

TABLE 5-2

COSTS FOR SOFTWARE DEVELOPMENT AND SYSTEMS INTEGRATION

	EFFORT (PM)	SCHEDULE (MONTHS)	FULL-TIME STAFF
SASET	574	27	21
SASET (CDR SPECIFIED)	646	45	14
SOFTCOST ADA	701	45	15
UH-1	247.5	45	5

closely approximated the actual schedule for the UH-1. This is significant because the actual schedule slipped a total of one year and six months when compared to the milestone dates specified in the RFP. A schedule summary shown in Appendix A compares the RFP date with actual dates for each milestone. One of the unanswered questions that arise when resulting schedule projections are compared to effort projections is the reason for the discrepancy between estimated and actual effort when schedule projections were very much on target with the actual schedule.

6.0 TRAINER QUALITY EVALUATION

There are various factors that are used to specify the types of quality desired in a particular software product. The class of software usually drives the quality factors that are emphasized as most important [7]. For example, if software is expected to have a long life cycle, then maintainability and expandability are rated as most important. If a software failure could result in the loss of human lives, software reliability, correctness, and testability would be emphasized. Quality indicators will vary depending upon the definition of a quality assessment framework.

There are two basic approaches for evaluating software quality 1) language specific and 2) non-language specific. Generally non-language specific methods focus more on the measure of software development techniques that promote quality (e.g. design techniques and methodology, design and code reviews) than do the language specific techniques. In addition to measuring the use of quality enhancing procedures, features of the actual software code are also measured. It is here that differences between language specific and non-language specific frameworks are most apparent. The non-language specific methods tend to measure generic aspects of the code such as the presence of machine code, excessive parameter passing, and global versus local data - in other words, the use of coding procedures proven to yield structured, descriptive, modular code showing high cohesion and low coupling. Conversely, the language specific approach measures the existence of features unique to the language that will enhance or detract from software quality. For example, Ada language features that enhance the quality of Ada code by promoting reusability include the use of generic packages, tasks, exceptions, and information hiding. In a language specific quality framework for Ada, it is these language features that would be measured.

Several specific methods and supporting tools were evaluated for potential application to the UH-1 FS project. Application of these tools and techniques are either manual or automated and most are not entirely objective. Subjectivity in software quality analysis is unfortunately somewhat inherent in the basic assumptions of what should be measured and how those measurements are made. Manual methods are numerous and varied, relying heavily on questionnaires and/or manual code analysis. These types of quality analyses are generally time intensive, not practically applicable to projects of moderate or large size, and not widely adopted.

With so many techniques available, the outlying question is "Which method/tool is the right one to use?" The advantage of any one approach over another is driven by the immediate project requirements and long term goals. For projects of moderate to large size (with respect to the measures being taken), an automated approach is obviously preferred. If specific features of the development language are of interest and the positive or negative impact of their use is considered important, then a language specific approach is warranted. These types of considerations address the immediate project requirements, but the long term goals with respect to the developer's software development system must also be considered. If software is primarily developed in one language, then a language specific approach may be preferred. If, however, any of several languages could be used or multiple languages are used within one project, then it may be impractical to acquire several language specific tools and try to integrate the results; a non-language specific technique may be preferable.

An automated language-specific technique was selected for the UH-1 FS program to support software quality evaluation - namely, AdaMAT/D (version 1.1). Supplementing the quality evaluation is an evaluation of data collected on error quantity and type. The following sections describe both approaches for measuring software quality.

6.1 ADAMAT/D RESULTS

The following sections present an overview of AdaMAT/D, a description of how it was applied to the UH-1 FS program, and subsequent results.

6.1.1 Product Overview

AdaMAT/D is an automated tool developed by Dynamics Research Corporation that operates by examining compilable Ada source code with respect to its quality assessment technique. The technique used by the tool is the counting of significant language features that are considered to promote or detract from the quality of the product. These counts are the metric elements. Metric element scores are shown as a ratio of the number of opportunities to comply with the preferred quality practice versus the number of actual compliances. For example, a metric score for proper declarations of constants would be calculated as the number of constants declared in the declarative section versus the quantity that could have been declared in the declarative section (as opposed to being hidden from the user in the code). The metric scores are then aggregated to a criteria level and then to a factor level. The factors evaluated by the tool are reliability, portability and maintainability. Seven criteria are evaluated: anomaly management, independence, modularity, self descriptiveness, simplicity, system clarity, and exactness. Criteria scores are derived from 250 metric values. The tool provides the capability to tailor the metrics gathered and to tailor the aggregation process; that is, the user has the ability to selectively omit metric elements and metrics. Weights can also be set to give greater importance to one metric over another or one criteria over another in the score calculations. Results can be viewed at any level in the hierarchy or reports can be triggered by user specified thresholds. Using thresholds, the user would indicate minimal acceptable scores and a report would be generated only if the scores were below the threshold.

Users were interviewed to obtain their opinion about the tool prior to procuring the tool for the UH-1 FS project. The questions asked focused on how the product is used and value of scores. All of the respondents felt it was difficult to learn how to use the tool at first but once it was made a part of the development cycle it became easier to apply. The major hurdle was educating the users on both the tool and the underlying metrics -- what they mean and how they work together to give a score. Almost all users examined the scores at the criteria level as opposed to the factor level. It was felt that pinpointing the cause for a low score and that the identification of areas where further Ada training would be beneficial was easier at this level.

6.1.2 Approach

AdaMAT/D is most effective when the tool is tailored to an organization's specific coding standards. AdaMAT/D is run on a module by module basis throughout implementation in order to detect areas of non-compliance to coding standards that detract unnecessarily from quality. Work is usually performed during the early stages of code development to provide ample time to review results and to implement changes prior to the start of the testing phase [8].

The first time user of AdaMat/D would apply the tool without any tailoring. The user would subsequently locate the code containing actual examples of non-adherence, analyze the code segments involved in order to determine the reason for non-adherence, the negative effects of non-adherence if any, and make sample modifications to the code to see the actual effects of obtaining adherence to the criteria. By a metric by metric analysis, the user would determine those data items to be collected from source code (when there is a good reason not to adhere to principal) and tailor the product accordingly [8].

When the application of the ADAMAT/D tool to the UH-1 FS redevelopment was discussed at a NSIA CWG meeting, one of the concerns that was raised was that different organizations would have different standard for coding, even within the same application domain, i.e., trainers. The concern was that if the tool were to be tailored for one organization, then subsequent projects would be required to conform to those same development standards.

We were unable to acquire interim deliveries of the source code throughout implementation. Our metric analysis started with the receipt of baseline version 15 of the source code which was the version of software shipped to the first training site in Los Alimitos, California, one month prior to Government Final Inspection. This version of the trainer was fully tested with the exception of the motion system. The source code contained 577 separate files or approximately 200,000 physical lines of code.

The AdaMat/D reporting mechanism allows you to create a report on a single file or for different combinations of Ada files. We had the option to calculate and report metric scores for each of the 577 files, however, given that the tool was being applied after-the-fact, the effort would not have yielded results that could be used to benefit the current project. Addressing the concerns voiced at the NSIA CWG meeting, it was decided that it would be beneficial to apply an untailored version of AdaMAT/D to several trainer applications and analyze the results prior to tailoring.

6.1.3 Results

Metric scores were calculated for three grouping of Ada source files and for the Ada source as a whole. The three groups were 1) application software, 2) support software (described in Table 4-2), and 3) software ported from the GAT and services software. Services software are general utilities that include math functions, string functions, data interpolation, graphics functions and conversion routines. Table 6-1 provides an overview of the resulting set of metric scores. Figure 6-1 provides a pictorial representation of the results. A report for each set of metric scores is contained in Appendix G.

Table 6-1 shows scores at the factor level for each software group. The scores are computed based on the number of opportunities to comply with the preferred quality practice versus the number of actual compliances. For example, there were a total of 98,225 opportunities for compliance to enhance reliability within application code of which there were 43,650 adherences. The results showed a high rating for portability with an overall score of .96 out of 1.0. Scores for reliability and maintainability were lower at .47 and .56 respectively. Individual metrics scores were evaluated to ascertain if there several attributes of the software that tended to pull the overall ratings downward. Of 108 metric elements applied in the application software area, 55 metric scores indicated a level of potential non-adherence below 70 percent. There were some design decisions that resulted in some of the lower ratings. For example, the trainer structural model emphasized a message passing scheme that resulted in a smaller proportion of hidden types. The impact that these decisions have to sustaining engineering tasks is unknown.

6.2 ERROR DENSITY ANALYSIS

Changes made to software during the development were formally reported on change report forms. Action requests were used during design and unit testing prior to the time that software was placed under formal configuration control. After unit and CSC testing, all changes were documented using Software Trouble Reports (STRs) and standard government Discrepancy Reports (DRs). If a government DR would result in a software change, then a STR would be generated. On each STR the developer would supply a description of the problem, when the problem occurred, the source and type of error, and all affected software and documentation.

TABLE 6-1

ADAMAT/D SCORES BY QUALITY CATEGORY

RELIABILITY

	Adherence	Non-Adherence	Total	Score
APPLICATION	43,650	54,575	98,225	0.44
SUPPORT	12,905	13,970	26,875	0.48
GAT & SERVICES	9,354	7,101	16,455	0.57
TOTAL	65,909	75,646	141,555	0.47

MAINTAINABILITY

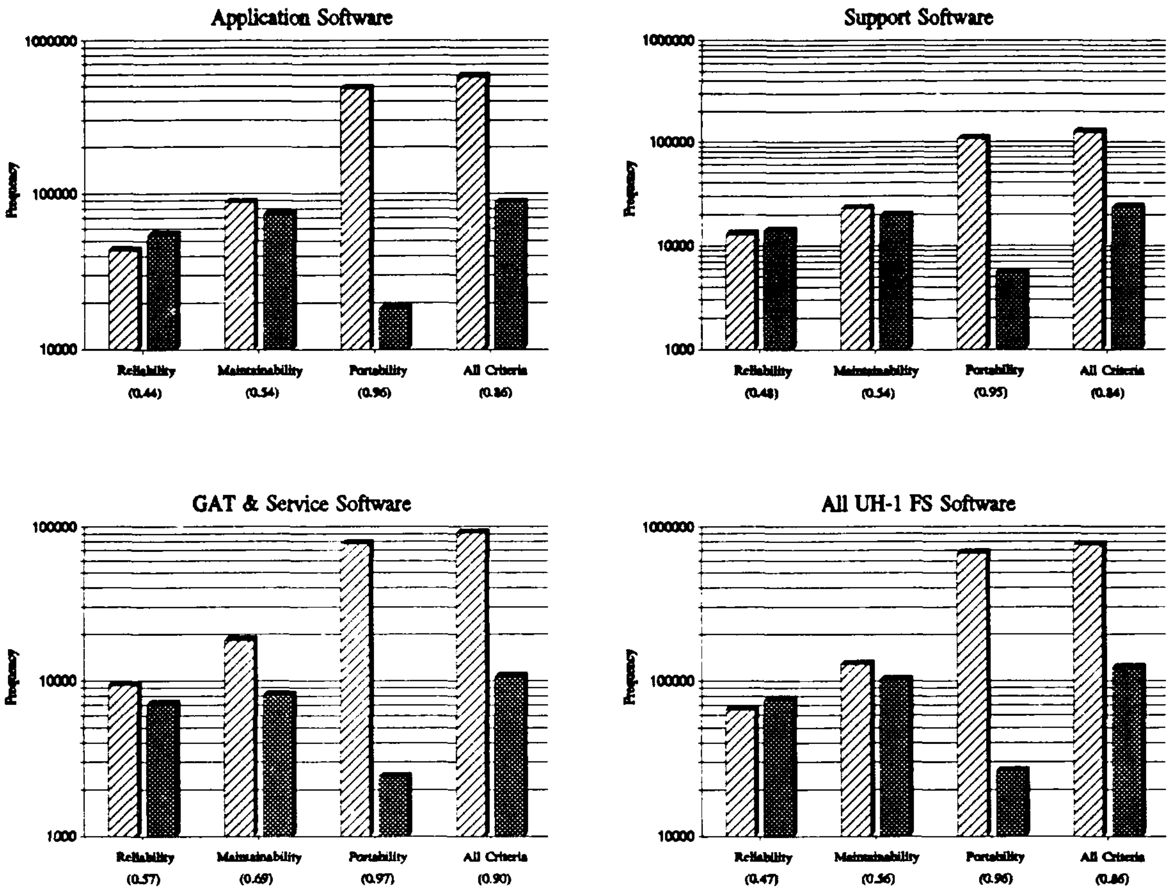
	Adherence	Non-Adherence	Total	Score
APPLICATION	87,611	74,012	161,623	0.54
SUPPORT	22,850	19,542	42,392	0.54
GAT & SERVICES	18,312	8,141	26,453	0.69
TOTAL	128,773	101,695	230,468	0.56

PORTABILITY

	Adherence	Non-Adherence	Total	Score
APPLICATION	487,577	18,361	505,938	0.96
SUPPORT	107,745	5,531	113,276	0.95
GAT & SERVICES	77,892	2,422	80,314	0.97
TOTAL	673,214	26,314	699,528	0.96

ADAMAT/D SCORES - ALL CRITERIA

	Adherence	Non-Adherence	Total	Score
APPLICATION	548,183	87,490	635,673	0.86
SUPPORT	124,568	23,568	148,136	0.84
GAT & SERVICES	91,112	10,609	101,721	0.90
TOTAL	763,863	121,667	885,530	0.86



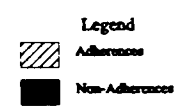


Figure 6-1. ADAMAT/D Results.

Reported errors are classified according to the source and type of error. The developer's form identified eight classifications and an additional category labelled "other". Figure 6-2 shows the classifications of STRs for the UH-1 FS project. The figure shows a significant number of STRs, i.e. 39 percent, in the performance problem category. The developer attributed the high number of performance problems to erroneous classification by project staff. There were few time critical problems on the UH-1 of the nature that would be described as a performance problem. When a problem was detected during testing, the tester would not necessarily know the source of the problem and mark the STR as a performance problem because the system did not "perform as expected". In retrospect, the developer suggested that the person who corrected the problem should have been the one to select the problem class.

Figure 6-3 shows the history of reported problems accumulated by month up to the time that the software was accepted at the first trainer site in August, 1990. The figure shows significant activity for a period of about three months and then gradually dropping off in the last four months of the project. To the developer's credit, the project sponsor was very impressed with the small number of open Discrepancy Reports at the end of the project. At the time the trainer completed its in-plant test at the developer's site, there were a total of only three open DRs. There was only one open DR when the trainer was installed and tested at the first training site which was quickly fixed.

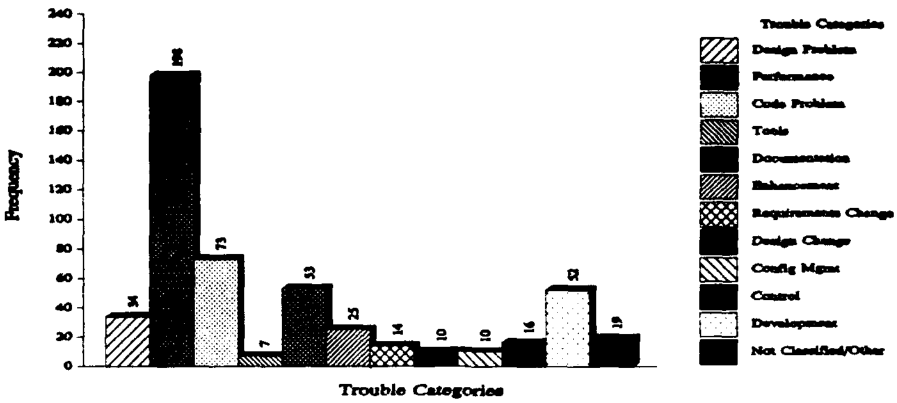


Figure 6-2. Trouble Report Classification.

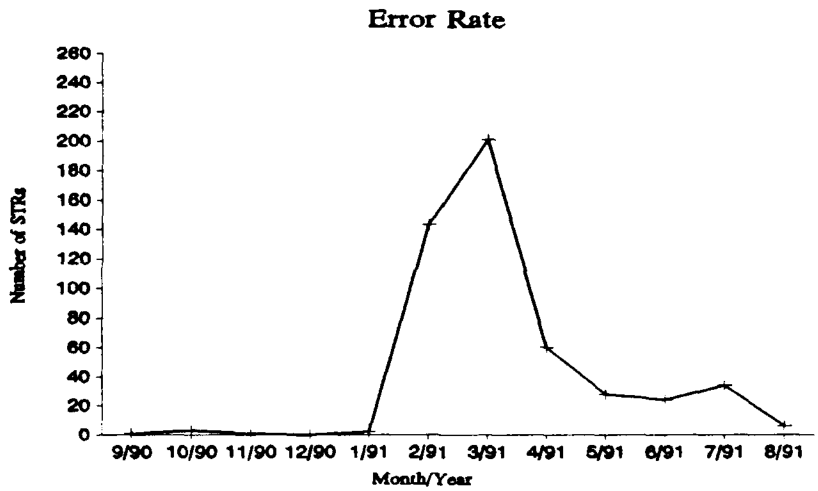


Figure 6-3. History of Reported Software Problems.

There were a total of 511 STRs generated on the UH-1 FS program. Software reliability, measured by the number of changes or error corrections made to the software is shown in Table 6-2.

TABLE 6-2

UH-1 FS ERROR/CHANGE RATE

ERRORS/KSLOC	2.38	SLOC = Physical Lines (includes comments)
	9.22	SLOC = Terminal Semicolons

7.0 CONCLUSION

Based on the early results of the UH-1 FS Ada Feasibility Study, the development team and those involved with the effort have concluded that Ada is a viable, usable technology capable of supporting real-time projects for training devices. The data collected on this project has led to a greater understanding of both the Ada language and its development methodologies. The study also raises some questions about the influence that other factors, i.e., structural model development methodology and redevelopments of existing systems, have on overall productivity. Additional data points are required to perform a more detailed analysis of the characteristics of Ada software development process in the trainer application domain. The following general observations were made by the study team during the experiment:

A SLOC count using the body semicolons counting convention was 60 percent larger than the terminal semicolons count. The body semicolons counting convention counts a statement terminated by a carriage return in the package specification and a terminal semicolon in the body of an Ada program. The size differential between body semicolons and terminal semicolons resulted from a packaging framework used on the UH-1 program which resulted in about three package specs for each body.

With the exception of tasking, the use of Ada features was comparable to SEL data. UH-1 results were compared with SEL trends in the use of Ada features over a span of seven years, beginning with their first Ada project in 1985. Four special Ada features were compared: strong typing, generics, package size, and tasking. The comparison indicates that the tasking feature of Ada is highly application dependent.

The phase distribution of effort was contrary to other published data [4] that indicates a shift of effort from the integration and test phases to design phases for Ada projects. Nearly 60 percent of the effort and schedule were expended after CDR. While additional data points are needed to validate these assumptions, the differences are attributed to four factors: 1) The UH-1 is a redevelopment of an existing system and this resulted in less time spent communicating requirements between the developer and the sponsor, 2) The developer utilized the concept of a domain specific software architecture (or structural model) which was developed on a previous IR&D project and applied to the UH-1, 3) UH-1 phase distribution included systems integration and testing at the first trainer installation site to obtain a fully functioning hardware-software system, 4) In an effort to meet PDR and CDR payment milestones for the developer, milestone dates were scheduled earlier than what may have otherwise been considered optimum.

A review of size estimates made at project milestones support the notions that 1) there is a tendency to underestimate support software, and 2) we tend to estimate in terminal semicolons as opposed to body semicolons. Size estimates were made in the proposal by the developer, at PDR by PM TRADE, and at CDR using function point analysis. In all cases estimates for support software were low by a factor of 40 percent or more. In all cases, estimates were comparable to the actual size base on the terminal semicolon counting convention.

Using a language expansion factor of 71 for Ada, the function point estimated sizes were high for both the ACS and the Real-Time CSCI. The estimate for the ACS had only a relative error of 11 percent as compared to the 32 percent relative error of the Real-Time CSCI. A comparison of the actual size of the Real-Time CSCI to the estimated size shows that an Ada language expansion factor of 48 would have yielded the correct results. Interpreting the guidelines [9] to define and count function point parameters, and extending the guidelines to training devices was not a straight-forward process. The greatest difficulty was determining how instrument display devices, malfunctions, and various flight controls should be grouped and counted.

In general, the model projections for effort were much higher than the actual effort expenditure reported by the developer. Model schedule projections closely approximated the actual schedule for the UH-1. It is believed that the high productivity experienced on the project can be attributed to two factors: 1) The UH-1 is a redevelopment of an existing system and this resulted in less time spent communicating requirements between the developer and the sponsor, 2) The developer utilized the concept of a domain specific software architecture (or structural model) which was developed on a previous IR&D project and applied to the UH-1. The cost models are typically used to estimate new development efforts.

The major obstacle in achieving useful results with AdaMAT/D is educating the users on both the tool and the underlying metrics -- what they mean and how they work together to give a score. The resulting set of metric scores were difficult to interpret at the factor level given that there were no historical data for comparison. It was decided that it would be beneficial to apply an untailored version of AdaMAT/D to several training devices and analyze the resulting trends prior to tailoring. The trend analysis could be used to develop coding guidelines for training devices.

APPENDIX A

SOFTWARE PROJECT DATA COLLECTION FORMS

FOR THE

UH-1 FS ADA FEASIBILITY PROJECT

PROJECT QUESTIONNAIRE

GENERAL INFORMATION

Please complete this form to the best of your ability for your project. If the question is not applicable, please mark it N/A. If you don't know the answer, leave it blank. Mark each page containing confidential or proprietary data "CONFIDENTIAL" on both its top and bottom in bold letters.

1. Your name: Katherine Miller Date: 11 / 20 / 91

2. Title: Software Engineer Phone: (301) 459 - 3711

3. Firm or Organization: IIT Research Institute

   Address: 4600 Forbes Blvd., Lanham, MD 20706

4. Name of Project: UH-1 FS Ada Systems Engineering Feasibility Project

5. Contract Number: N61339-88-C-0010

6. Customer Name: Naval Training System Center

7. Project Overview Description: This project is a redevelopment of an existing UH-1 Flight Simulator from assembly to Ada to improve its capability to be supported and to provide a means to split a single 4-cockpit trainer into two 2-cockpit trainers, if needed. This project questionnaire reflects the following developmental software components: IOS CSC, Trainee Station CSC, DCE Diagnostics, Target Computer Diagnostics, ACS CSC.

8. Developer Contact: Ron Murphy Phone: (516) 563 - 7940

9. Customer Contact: Robert Paulson Phone: (407) 380 - 4362

10. Current Status: First trainer delivered and accepted 9/91. Source lines of code counts provided in this form were obtained from the "Cold Start" tape which was cut in 7/91. ACS delivered and accepted in 9/91.

PROJECT QUESTIONNAIRE

GENERAL INFORMATION

1. **System/Software Characteristics**

   a. **Operating Environment** (check one):

[ ]	Manned Flight	[ ]	Unmanned Flight
[ ]	Avionics	[ ]	Shipboard/Submarine
[x]	*Ground*	[ ]	Commercial

(SASET: Class of Software)

   b. **Applications domain**:

[ ] Automation
[ ] Command & Control
[ ] Telecommunications
[ ] Test Systems
[x] Simulation
[ ] Data Processing
[ ] Environment/Tools
[ ] Scientific
[ ] Avionics
[ ] *Other*__________________

(SoftCost-Ada: Type of Software)

2. **Complexity**

   a. **Rate the difficulty of the processing logic:**

[ ] Very low - Strait line code. Standard types. General structures. Simple math. No tasking.
[ ] Low - Simple operators. Standard types. General structures. Simple math. Simple data manipulation. No tasking.
[ ] Nominal - Strait forward logic. Generics and standard structures. Standard I/O. Simple tasking.
[x] High - Highly nested logic. Numeric types. Libraries of packages and generics. Complicated I/O. Concurrent tasking.
[ ] Very high - Stochastic logic. Unique types. Libraries of packages, tasks, and generics. Sophisticated math and I/O. Rendezvous.
[ ] Extra high - Dynamic resource allocation. Unique types. Special libraries. Time dependent task scheduling. Multiple exception handlers. Optimization and efficiency concerns.

(SoftCost-Ada: Product Complexity)

PROJECT QUESTIONNAIRE

GENERAL INFORMATION

b. The complexity of this CSCI is best characterized by which of the following statements?:

[ ] Very low - Straightline code with a few non-nested structured programming operators: DOs, CASEs, IF-THEN-ELSEs. Simple predicates. Evaluation of simple expressions: for example, A=B+C*(D-E). Simple read, write statements with simple formats. Simple arrays in main memory.

[ ] Low - Straightforward nesting of structured programming operators. Mostly simple predicates. Evaluation of moderate level expressions, for example D=SQRT (B**2-4.*A*C). No cognizance needed of particular processor or I/O device characteristics. I/O done at GET/PUT level. No cognizance of overlap. Single file subsetting with no data structure changes, no edits, no intermediate files.

[ ] Nominal - Mostly simple nesting. Some intermodule control. Decision tables. Use of standard math and statistical routines. Basic matrix and vector operations. I/O processing includes device selection, status checking and error processing. Multifile input and single file output. Simple structural changes, simple edits.

[x] High - Highly nested structured programming operators with many compound predicates. Queue and stack control. Considerable intermodule control. Basic numerical analysis: multi-variate interpolation, ordinary differential equations. Basic truncation, roundoff concerns. Operations at physical I/O level (physical storage address translations; seeks, reads, etc). Optimized I/O overlap. Special purpose subroutines activated by data stream contents. Complex data restructuring at record level.

[ ] Very high - Reentrant and recursive coding. Fixed-priority interrupt handling. Difficult but structured numerical analysis: near-singular matrix equations, partial differential equations. Routines for interrupt diagnosis, servicing, masking. Communication line handling. A generalized, parameter-driven file structuring routine. file building, command processing, search optimization.

[ ] Extra high - Multiple resource scheduling with dynamically changing priorities. Microcode-level control. Difficult and unstructured numerical analysis: highly accurate analysis of noisy, stochastic data. Device timing-dependent coding, microprogrammed operations. Highly coupled, dynamic relational structures. Natural language data management.

(Ada COCOMO: Software Product Complexity)

c. Degree of Real-time

[ ] Low - No tasking; essentially batch response
[ ] Nominal - Interactive with limited Ada tasking
[x] High - Interrupt driven with tasking in milliseconds
[ ] Very high - Concurrent tasking with rendezvous in milliseconds
[ ] Extra high - Concurrent tasking with rendezvous in nanoseconds

(SoftCost-Ada: Degree of Real-Time)

PROJECT QUESTIONNAIRE

GENERAL INFORMATION

3. **Reliability**

a. **Effect of a software failure**

[ ] Inconvenience
[x] Easily-recoverable loss
[ ] Loss of human life
[ ] Moderate loss
[ ] Major financial loss

(Ada COCOMO: Required Software Reliability)

4. **Interfaces**

a. **Man-machine Interaction.** Address the level of man interaction inherent in the system.

[ ] Extensive and complex interactive type system
[x] Highly interactive system
[ ] Small level of interaction with system - system operates mostly in an autonomous fashion
[ ] System is almost fully autonomous

(SASET: Man Interaction)

b. **Software Interface Complexity:**

How many other software systems and peripheral communications equipment with various protocols and baud rates does this software system interface with? 8

(Note: Counted as 6 HWCI standard peripherals (disk, console, tape, printer, LAN, voice system) plus (1) IOS indicators and controls and (1) Trainee Station indicators and controls.)

(SASET: Software Interfaces)

5. **Software Testability**

[ ] Very difficult
[x] Difficult
[ ] Time intensive
[ ] Easy

(SASET: Software Testability)

PROJECT QUESTIONNAIRE

GENERAL INFORMATION

6. Reused Code

a. Select the intended use of the majority of the software packaged for reuse:

[ ] Not for reuse elsewhere
[ ] Reuse within single-mission products
[x] Reuse across single product line
[ ] Reuse in any application

(Ada COCOMO: Required Reusability)

b. Reuse Costs

[x] Low - No reuse library. Limited packaging for future reuse
[ ] Nominal - Reuse library employed. Less than 10% of software being packaged for reuse.
[ ] High - Reuse library being populated. Less than 20% of software being packaged for future reuse.
[ ] Very High - Reuse library exploited. More than 20% of software being packaged for future reuse.

(SoftCost-Ada: Reuse Costs)

PROJECT QUESTIONNAIRE

DEVELOPMENT METHODOLOGY

1. Milestones

a. Schedule

Milestone	Expected Date	Actual Date
Project Start Date	12/ 1/87	12/ 1/87
System Requirements Review	1/ /88	1/20/88
Software Specification Review	4/ /88	4/27/88
System Design Review	N/A	N/A
System Hardware PDR	N/A	N/A
System Software PDR	9/ /88	10/ 4/88
System Software CDR	3/ /89	7/24/89
Test Readiness Review	11/ /89	3/ /91
Functional Configuration Audit	1/ /90	3/ /91
Physical Configuration Audit	2/ /90	3/ /91
Formal Qualification Review	2/ /90	N/A
Operational Test and Evaluation	3/ /90	4/ /91
Project Completion Date	3/ /90	9/ /91

(Note: The original milestone schedule shown under the expected date column was specified in the RFP. Difference between the expected and actual dates may have been caused by two major contract modifications: one for the ACS and one regarding the use of the Navy device TH-11.

There were two CDRs held. The first CDR held on 7/24/89 was for the real-time CSCI plus DCE and target computer diagnostics. The second CDR held on 11/28/89 was for the ACS CSC.

Because of differences in terminology between contract performance milestones and the terminology noted above, the following assumptions were made to designate expected dates:

1. *Government Final Inspection Complete in the CDRL coincides with Formal Qualification Review.*

2. *The dates for the Reliability Test and Maintainability Demo in the CDRL coincides with Operational Test and Evaluation*

3. *Functional Configuration Audit {at Ft. Rucker} is scheduled 60 days prior to Project completion date (Scheduled Government Acceptance).*

4. *Physical Configuration Audit is conducted {at Ft. Rucker} at the beginning of Government Final Inspection.)*

5. *Project completion date coincides with the end of Government Final Inspection at the first training site in Los Alimitos, CA. Government Final Inspection of the ACS was conducted at Ft. Rucker during 1/92.*

(SASET: Schedule)

PROJECT QUESTIONNAIRE

DEVELOPMENT METHODOLOGY

b. Percent of development schedule devoted for Preliminary Design phase

[ ] 40 [ ] 33 [ ] 25 [ ] 17 [x] 10 [ ] 5

(Note: Five months out of 41, 12 %, were spent on preliminary design.)

(Ada COCOMO Σ Factor: Risk Elimination By PDR)
(Ada COCOMO Σ Factor: Design Thoroughness By PDR)

2. Development Standards

a. Check all types of standard used in this development:

[ ] None
[ ] Ada Programming Standards
[ ] Commercial
[ ] IEEE
[x] Military
[ ] Other

(SoftCost-Ada: Degree of Standardization)

b. List the name(s) of these standard(s): MIL-STD-2167, MIL-STD-2167A (for SDD only)

(SASET: Software Documentation)

c. Were these standards tailored specifically for use on this effort?

[ ] Yes [x] No

(SoftCost-Ada: Degree of Standardization)
(SASET: Software Documentation)

d. List the name(s) of the software documents required: SDP, SRS, STP, SDD, MMR, TTPRR, CSOM (Computer Systems Operator's Manual), VDD, CRISD, SPS

(SASET: Software Documentation)

3. Risk Management

a. Number and criticality of risk items

[x] $\leq$ 5, Noncritical

DEVELOPMENT METHODOLOGY

[ ] > 5, Noncritical
[ ] 1, Critical
[ ] 2-4, Critical
[ ] 5-10, Critical
[ ] > 10, Critical

(Ada COCOMO Σ Factor: Risk Elimination By PDR; not required by COSTMODL)

b. Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR

[ ] Fully
[ ] Mostly
[ ] Generally
[ ] Some
[ ] Little
[x] None

(Ada COCOMO Σ Factor: Risk Elimination By PDR)

c. Schedule, budget, and internal milestones through PDR compatible with Risk Management Plan

[ ] Fully
[ ] Mostly
[ ] Generally
[ ] Some
[ ] Little
[x] None

(Ada COCOMO Σ Factor: Risk Elimination By PDR)
(Ada COCOMO Σ Factor: Design Thoroughness By PDR)

d. Tool support available for resolving risk items

[ ] Full
[ ] Strong
[ ] Good
[x] Some
[ ] Little
[ ] None

(Ada COCOMO Σ Factor: Risk Elimination By PDR)

4. Software Reviews

a. Select all informal reviews held on the software during this development:

DEVELOPMENT METHODOLOGY

- [ ] None
- [x] Quality inspections/audits
- [x] Design walkthroughs
- [x] Design inspections
- [x] Code walkthroughs
- [x] Code inspections
- [ ] Other ______________________

(SoftCost-Ada: Use of Peer Reviews)

b. Select all management reviews held on the software for this project:

- [ ] None
- [x] Monthly project reviews
- [x] Weekly status reviews
- [ ] Other ______________________

(SoftCost-Ada: Use of Peer Reviews)

5. System/Software Requirements

a. Select the option which corresponds to the level of definition and understanding of system requirements:

- [ ] Very little definition and understanding of system requirements
- [ ] Questionable definition and understanding of system requirements
- [x] Fairly complete definition and understanding of system requirements
- [ ] Very complete definition and understanding of system requirements

(SASET: System Requirements)

b. Select the option which corresponds to the level of definition and understanding of software requirements:

- [ ] Very little definition and understanding of software requirements
- [ ] Questionable definition and understanding of software requirements
- [x] Fairly complete definition and understanding of software requirements
- [ ] Very complete definition and understanding of software requirements

(SASET: Software Requirements)

c. How will overall technology changes impact the project?

- [ ] During the development, the requirements will change more than once to upgrade the system, due to significant improvements in technology
- [ ] During the development, there will be at least one (but less than three) significant

DEVELOPMENT METHODOLOGY

modifications to the system due to technology upgrades
- [x] During the development there will be some minor modifications due to technology upgrades
- [ ] There will be no changes to the system or requirements during the development effort.

(Note: The target computer changed from MC68020 to MC68030. The toolset is constantly evolving.)

(SASET: Technology Impacts)

d. Select the percentage of software requirements well established:

[ ] >90% [x] >60% [ ] >50% [ ] >30% [ ] <30%

(SoftCost-Ada: Requirements Volatility)

e. System requirements baselined, under rigorous change control

- [x] Fully
- [ ] Mostly
- [ ] Generally
- [ ] Some
- [ ] Little
- [ ] None

(Ada COCOMO Σ Factor: Requirements Volatility)

f. Level of uncertainty in key requirements areas: mission, user interface, hardware, other interfaces

- [ ] Very little
- [x] Little
- [ ] Some
- [ ] Considerable
- [ ] Significant
- [ ] Extreme

(Ada COCOMO Σ Factor: Requirements Volatility)

g. Organizational track record in keeping requirements stable

- [ ] Excellent
- [x] Strong
- [ ] Good
- [ ] Moderate
- [ ] Weak
- [ ] Very Weak

PROJECT QUESTIONNAIRE

DEVELOPMENT METHODOLOGY

(Ada COCOMO Σ Factor: Requirements Volatility)

h. Use of incremental development to stabilize requirements

[ ] Full
[ ] Strong
[ ] Good
[x] Some
[ ] Little
[ ] None

(Ada COCOMO Σ Factor: Requirements Volatility)

i. System architecture modularized around major sources of change

[ ] Fully
[ ] Mostly
[ ] Generally
[x] Some
[ ] Little
[ ] None

(Ada COCOMO Σ Factor: Requirements Volatility)

j. Level of uncertainty in key architecture drivers: mission, user interface, hardware, COTS, technology, performance

[ ] Very Little
[ ] Little
[x] Some
[ ] Considerable
[ ] Significant
[ ] Extreme

(Ada COCOMO Σ Factor: Design Thoroughness By PDR)

6. Commercial off-the-shelf software (COTS)

a. Select the option which best describes the expected impact of integrating commercial off-the-shelf software into the system:

[ ] There will be many impacts on the design/development effort to ensure that the vendor supplied COTS software will interface correctly with the developed operational software.
[x] There will be some impacts on the design/development effort to ensure that the vendor supplied COTS software will interface correctly with the developed operational software.
[ ] There will be few impacts created by the COTS software packages to support the

DEVELOPMENT METHODOLOGY

operating environment of the applications software.

[ ] There will be no impacts caused by the purchased software as the purchased software only performs an operating environment support function (i.e., operating system).

(SASET: COTS Software)

7. Use of Software Tools

a. Specify the type of environment that will be used to develop the software:

[ ] Basic Ada language tools
[x] MAPSE, plus access to host tools
[ ] MAPSE, plus access to host/target tools
[ ] Full, life cycle APSE
[ ] APSE

(Note: Tools include compiler, library manager, editor, linker/loader, CCC, Harvard Project Manager)

(SASET: Software Development Tools)
(SoftCost-Ada: Use of Software Tools/Environment)

b. Specify the type of tools that will be used to develop the software:

[ ] Basic microprocessor tools
[ ] Basic minicomputer tools
[x] Strong mini, Basic maxicomputer tools
[ ] Strong maxi, MAPSE
[ ] Advanced maxi, APSE

(Ada COCOMO: Use of Software Tools)

c. Tool support for developing and verifying Ada package specs

[ ] Full
[ ] Strong
[ ] Good
[x] Some
[ ] Little
[ ] None

(Ada COCOMO Σ Factor: Design Thoroughness By PDR)

8. Use of Modern Programming Practices

a. Degree to which modern programming practices are used in developing software:

[ ] No use
[ ] Beginning
[x] Reasonably experienced in some

PROJECT QUESTIONNAIRE

DEVELOPMENT METHODOLOGY

[ ] Reasonably experienced in most
[ ] Routine use of all

(Ada COCOMO: Use of Modern Programming Practices)
(Ada COCOMO Σ Factor for Maintenance Model: Use of MPPs)

b. Ada Development Methodology

[ ] Structured programming
[x] Object-oriented design plus structured programming
[ ] Ada packaging methods
[ ] Integrated life-cycle methodology which exploits Ada reusability concepts
[ ] Other ______________________

(SoftCost-Ada: Use of Modern Software Methods)

c. Maintenance Conformance to the Ada Process Model

[ ] Full
[ ] General
[x] Often
[ ] Some
[ ] Little
[ ] None

(Ada COCOMO Σ Factor for Maintenance Model: Conformance)

PROJECT QUESTIONNAIRE

SOFTWARE SIZE

1. **Size Estimates**

   a. **Number of CSCIs:** 2

   (SASET: Number of CPCIs)

   b. **Identify counting convention which is used to provide requested sizing information in (c).**

   [ ] Physical lines
   [x] Terminal semicolons
   [x] Body semicolons
   [ ] Other ________________
   [ ] Non-comment, non-blank lines
   [ ] Essential semicolons

   c. **Enter the requested sizing information below, in thousands of lines of source code (KSLOCs).**

   UH-1 FS Source Code: By Terminal Semicolons

Deliverables Program	Language	New	Reused/ Modified	Reused/ Unmodified	Software Type
IOS CSC	Ada	11188	483		Application
Trainee Station CSC	Ada	11556	933		Application
ACS CSC	Ada	6386	137		Application
	Ada	1701	72		Support
Common Code	Ada	2892		3267	Application
	Ada	15	82		Support
	DCL	469	115		
	Assembly	6545			
DCE Diagnostics	Ada	6804	530		Support
Target Computer Diagnostics	Ada	604	457		Support
Daily Readiness	Ada	908	279		Support
TOTAL		**49,068.00**	**3,088.00**	**3,267.00**	**55,423.00**

PROJECT QUESTIONNAIRE

SOFTWARE SIZE

UH-1 FS Source Code: By Body Semicolons

Deliverables Program	Language	New	Reused/ Modified	Reused/ Unmodified	Software Type
IOS CSC	Ada	15021	570		Application
Trainee Station CSC	Ada	17789	933		Application
ACS CSC	Ada	9768	137		Application
	Ada	1787	72		Support
Common Code	Ada	15642		9997	Application
	Ada	15	82		Support
	DCL	469	115		
	Assembly	6545			
DCE Diagnostics	Ada	6804	677		Support
Target Computer Diagnostics	Ada	3017	544		Support
Daily Readiness	Ada	908	345		Support
TOTAL		**77,765.00**	**3,475.00**	**9,997.00**	**91,237.00**

(SoftCost-Ada: Ada Usage Factor)
(SoftCost-Ada: New, Reused, Modified Ada Components)
(SoftCost-Ada: New, Reused, Modified Other Components)
(SASET: Primary Software Language)
(SASET: Programing Language)
(SASET: Direct Input for SLOC)

d. **Reused software:** 11 %

(SoftCost-Ada: Reuse Benefits)

e. **Number of delivered source instructions adapted from existing software to form the new product:** 13.47 KSLOC

% of adapted software's design modified in order to adapt it to new environment: 10 %

% of adapted software's code modified in order to adapt it to new environment: 30 %

SOFTWARE SIZE

% of effort required to integrate the adapted software into the new product and to test the resulting product as compared to the normal amount of integration and test effort for software of comparable size: 10 %

(Ada COCOMO: Adapted Code)

2. Database Size

a. **Database size** *(in bytes or characters as percentage of total program size)*: 5 %

(Ada COCOMO: Database Size)
(SoftCost-Ada: Database Size)

PROJECT QUESTIONNAIRE

PROJECT STAFFING

1. **Staff Size/Availability**

a. **Staff availability:** 45 %

(SoftCost-Ada: Staff Resource Availability)

b. **Percent of required top software architects available to project**

[ ] 120 [ ] 100 [ ] 80 [x] 60 [ ] 40 [ ] 20

(Ada COCOMO Σ Factor: Risk Elimination By PDR)
(Ada COCOMO Σ Factor: Design Thoroughness BY PDR)

c. **Difficulty of staffing due to special training and clearances:**

[ ] Staffing of the project will be difficult because of special training or security requirements.
[ ] Initial staffing will be difficult because of special training or security requirements.
[x] Staffing of the project is projected to be fairly easy but there are some training requirements.
[ ] Staffing will not pose any problem at all.

(SASET: Personnel Resources)

2. **Staff Skill/Experience**

a. **Skill Level of Analysts**

[ ] Bottom 15% [ ] 35% [ ] 55% [x] 75% [ ] Top 90%

(Ada COCOMO: Analyst Capability)
(SoftCost-Ada: Analyst Capability)

b. **Skill Level of Programmers**

[ ] Bottom 15% [ ] 35% [x] 55% [ ] 75% [ ] Top 90%

(Ada COCOMO: Programmer Capability)

c. **Average experience with similar applications:** 2 years, 0 months

(Ada COCOMO: Applications Experience)
(SoftCost-Ada: Applications Experience)

d. **Average level of virtual machine experience of the project team developing the software module:**

PROJECT STAFFING

_2_ years _0_ months

(Ada COCOMO: Virtual Machine Experience)

e. Host Machine Expertise:

[ ] Inexperienced - completely new hosting hardware system
[x] Little experience - mostly new hosting hardware system
[ ] Average experience - most of the hardware system has been utilized by members of the development team before
[ ] Highly experience - extensive experience with hardware system

(SASET: Hardware Experience)

f. Software Language and Operating System Expertise:

[ ] Completely new hosting operating system or software language
[x] Few people with experience with operating system and/or software language
[ ] The software language and operating system have been utilized by the company before
[ ] Extensive experience with the software language and operating system

(SASET: Software Experience)

g. Experience with chosen development methodology:
_2_years _0_months

(SoftCost-Ada: Ada Methodology Experience)

h. Experience with Ada Process Model

[ ] Successful on > 1 mission critical project
[ ] Successful on 1 mission critical project
[ ] General familiarity with practices
[ ] Some familiarity with practices
[x] Little familiarity with practices
[ ] No familiarity with practices

(Ada COCOMO: Experience with Ada Process Model)

i. Project team's equivalent duration of experience (at the beginning on the project/build) with the programming language to be used:

_0_years _6_months

(Ada COCOMO: Programming Language Experience)
(SoftCost-Ada: Ada Language Experience)

PROJECT STAFFING

j. **Number of Ada projects completed by team members:** 0

(SoftCost-Ada: Number of Ada Projects Completed)

k. **Ada environment experience:**

[x] Less than 3 months of experience
[ ] Between 3 - 6 months of experience
[ ] Between 6 - 12 months of experience
[ ] Over 1 year of experience

(SoftCost-Ada: Ada Environment Experience)

l. **Level of product familiarity of the development team:**

[ ] This application is a new project not in our current line of business
[ ] This application is a normal development project that is a part of our current line of business
[x] This application is a familiar type of project having already been developed by the company before or similar to other projects we have developed
[ ] Many applications of this type have been developed by the company (greater than 7)

(SASET: Development Team)

3. **Teamwork Capability**

a. **Select the type of team used for software development:**

[ ] Design teams
[x] Interdisciplinary teams
[ ] Not used
[ ] Programming teams
[ ] Participatory teams

(SoftCost-Ada: Team Capability)

PROJECT QUESTIONNAIRE

COMPUTER SYSTEM

1. **Development Environment**

a. Number of different types of workstations: 2 (0 to 100)

(SASET: Workstation Types)

b. Rate the virtual machine volatility of the development system, based on frequency of major/minor changes:

[x] 12 months (major) / 1 month (minor)
[ ] 6 months (major) / 2 weeks (minor)
[ ] 2 months (major) / 1 week (minor)
[ ] 2 weeks (major) / 2 days (minor)

(Ada COCOMO: Virtual Machine Volatility - Host)

c. Select the following option that best assesses the embedded features of the development system:

[ ] Hardware is to be developed, but its completion will occur long before the software is to be ready
[ ] Hardware is to be developed on the contract, it is to be developed concurrently with the software and the hardware can/does have major impacts on the software
[ ] Hardware is to be developed on the contract, it is to be developed concurrently with the software but the hardware has little impact on the software
[x] No new hardware is to be developed under the effort; there will be no impact on the software development

(SASET: Embedded Development System)

d. Rate the software tool/environment stability of the development system:

[ ] Very Low - Buggy compiler. APSE change every 2 weeks.
[ ] Low - Stable but incapable compiler. APSE change every month. New tool rate 1 per week.
[ ] Nominal - Stable compiler. APSE change every 3 months. New tool rate 1 per quarter.
[ ] High - Stable compiler. APSE change every 4 months. New tool rate 1 per month.
[x] Very High - Stable compiler capable of tasking. APSE change every 6 months. New tool rate 1 per quarter.
[ ] Extra High - Stable and fully capable compiler. APSE change ever 6 months. New tool rate 1 per 6 months.

(SoftCost-Ada: Software Tool/Environment Stability)

e. Address the difference between the development hardware system and the host system:

[ ] Development computer significantly different than target computer, hardware emulation

COMPUTER SYSTEM

or math modelling required for missing hardware of software.

[ ] Development computer different than target computer, some hardware emulation or math modeling may be required for missing hardware or software.

[x] Some elements of the hardware/software development system are different from the target system but no problems or modifications are foreseen.

[ ] Development and target hardware/software system are identical or are one in the same.

(SASET: Development Versus Host System)

2. Target Computer Configuration

a. **Rate the virtual machine volatility of the target system, based on number of major/minor changes:**

[x] 12 months (major) / 1 month (minor)
[ ] 6 months (major) / 2 weeks (minor)
[ ] 2 months (major) / 1 week (minor)
[ ] 2 weeks (major) / 2 days (minor)

(Ada COCOMO: Virtual Machine Volatility-Target)

b. **Identify the system architecture:**

[ ] Centralized (single processor)
[ ] Tightly-coupled (multiple processor)
[x] Loosely-coupled (multiple processor)
[ ] Federated (Functional processors communicating via a bus)
[ ] Distributed (centralized database)
[ ] Distributed (distributed database)

Number of processors: 6

(SoftCost-Ada: System Architecture)
(SASET: Hardware System Type)

3. Performance Requirements

a. **Main Storage Constraint:** < 50 %

(Ada COCOMO: Main Storage Constraint)
(SASET: Percent of Core Utilized)

b. **Overall Hardware Constraints.** *Overall hardware refers to processor memory, I/O capacity, and throughput (i.e. CPU speed) available within the target computer system.*

COMPUTER SYSTEM

- [ ] Close to 100% utilization
- [ ] Difficult hardware capacity limitations (85% to 95%)
- [ ] Average hardware capacity limitations (75% to 85%)
- [ ] Minimal hardware capacity limitations (50% to 75%)
- [x] Less than 50% of available processor resources

(SASET - Hardware Constraints)
(SoftCost-Ada: Degree of Optimization)

c. **Execution Time Constraints.** *Select the percentage which best reflects the percentage of available execution time expected to be used by the subsystem and any other subsystems consuming the execution time resource.*

[x] at most 50%   [ ] 70%   [ ] 85%   [ ] 95%

(Ada COCOMO: Execution Time Constraint)

d. **Select the criteria which reflects the performance constraints of the software system:**

- [ ] Mission critical, error free or very difficult response times (real-time software)
- [x] High reliability or difficult response times
- [ ] Average reliability (non real-time software)
- [ ] Non-critical software with no tight performance requirements

(SASET: Timing and Criticality)

4. Microprocessor Code

a. **Percentage of software functions that are to be implemented in firmware:** < 5 %

(Note: Bootstrap and downloading functions were partially implemented in firmware.)

(SASET: Percentage of Microprocessor Code)

PROJECT QUESTIONNAIRE

DEVELOPMENT ENVIRONMENT

1. **Project Organization**

   a. **Number of organizations within the company significantly involved during the software development:** 5

   (SoftCost-Ada: Number of Organizations)

   b. **Scope of Support**

   [x] Low - No support to non-software organizations
   [ ] Nominal - Liaison support to non-software organizations
   [ ] High - Extensive support to system test organizations
   [ ] Very High - Extensive support to system engineering and test organizations. CSSR/CSCSC reporting requirements.

   (SoftCost-Ada: Scope of Support)

   c. **Organizational Interface Complexity**

   [ ] Single costumer collocated with developer
   [ ] Single customer, single interface
   [ ] Multiple internal interface, single external interface
   [x] Multiple internal and external interfaces
   [ ] Multiple interfaces, geographically distributed

   (SoftCost-Ada: Organizational Interface Complexity)

   d. **Number of locations at which software is developed (from 1 to 100):** 1

   (SASET: Development Locations)

   e. **Number of customer locations:** 5

   (Note: NTSC, FL; Ft. Rucker, AL; AVSCOM, St. Louis; Peoria, IL; and Los Alamitos, CA)

   (SASET: Customer Locations)
   (SASET: Information Travel Requirements)

2. **Computer Resources**

   a. **Characterize the development facilities and the perceived availability of the hardware (terminals and computers):**

   [ ] Development will be restricted due to hardware unavailability caused by high utilization or special hardware needs.
   [ ] Development is to occur on shared hardware that has varied utilization but generally

utilization is high (hardware shared by more than one project)

[ ] Development is to occur on hardware shared between a small group of projects: hardware availability is generally good.

[x] Development is to occur on hardware dedicated to the project and hardware availability is excellent

(SASET: Development Facilities)

b. Computer resource availability

[ ] Extreme equipment and facility limitations
[ ] Computer shared or remotely accessible
[x] Interactive access to dedicated computer resources
[ ] Dedicated facilities with multiple LAN-servers/worker
[ ] Software factory with multiple LAN-servers and specialized Ada machines

(SoftCost-Ada: Computer Resource Availability)

c. Select the average time required to submit a job to be run until the results are back in the developer's hand:

[x] Interactive, 1 terminal/person [ ] Interactive, .3 terminal/person

[ ] < 4 hours [ ] 4 - 12 hours [ ] > 12 hours

(Ada COCOMO: Computer Turnaround Time)

3. Security and Privacy Restrictions

a. Classified Application:

[x] Unclassified
[ ] Classified (Secret, Top Secret)

(Ada COCOMO: Classified Security Application)

b. Security Requirements

[x] None
[ ] Database integrity/privacy considerations
[ ] Physical security with access controls
[ ] Demonstrably correct trusted system. Physical security with access controls.
[ ] Verifiably correct trusted system. Physical security with access controls.

(SoftCost-Ada: Security Requirement)

DEVELOPMENT ENVIRONMENT

c. **Internal Computer System Security Safeguards**

[x] None
[ ] Security policy well defined and enforced
[ ] Marking - Access control labels are associated with all data
[ ] Identification - Access is based on who is accessing and the levels of information that the subject is authorized to access.
[ ] Accountability - Audit information is kept and protected. Actions affecting security can be traced to responsible party
[ ] Assurance - System contains trusted mechanisms that are independently evaluated to provide assurance that the system is accountable
[ ] Continuous - The mechanisms that provide assurance are continuously protected against tampering and unauthorized changes

(SASET: Software Security)

PROJECT QUESTIONNAIRE

RESOURCE ALLOCATION

1. **Effort**

   a. **Total Staff:** 247.5 (staff months effort end-to-end based on 152 hours/staff month)

   b. **Minimum Staff Size:** ____

   c. **For each software activity, please provide the total effort, by phase, in staff-months it took to complete.**

Phase	WBS # Primary	WBS Task Primary	WBS # Secondary	WBS Task Secondary	Hours Used
I	2171	Real Time	0000	Sys Requirements	99.50
			1000	S/W Requirements	677.00
			2000	Top-Level Design	427.00
	2172	Non-Real Time	0000	Sys Requirements	37.00
			1000	S/W Requirements	211.00
			2000	Top-Level Design	531.50
		ECP - ACS (6)	9006		8.00
	2311	Phase I Eng Data			1,177.00
	2235	Reviews			118.00
	2191	Benchmark Testing			308.00
	2211	Sys Engineering			2,465.00
	2234	Quality Assurance			28.00
				Total Hrs Phase I	6,087.00

Phase	WBS # Primary	WBS Task Primary	WBS # Secondary	WBS Task Secondary	Hours Used
II	2451	Real Time	3000	Detailed Design	2,799.50
			9006		4.00
	2452	Non-Real Time	3000	Detailed Design	2,713.50
			9006		59.50
	2510, 2573	Engineering Data			724.50
	2485	Reviews			150.00
	2613	Benchmark Testing			660.00
	2563	Phase II Eng Data			1,550.00
	2900	ECP (3-6)	0000		32.00
	2900	ECP (3-6)	2451		24.00
	2900	ECP (3-6)	2452		231.00
	2900	ECP (3-6)	2510		2.00
	2484	Quality Assurance			403.00

PROJECT QUESTIONNAIRE

RESOURCE ALLOCATION

Total Hrs Phase II	9,353.00

Phase	WBS # Primary	WBS Task Primary	WBS # Secondary	WBS Task Secondary	Hours Used
III	2761	Real Time	4000	Implementation	1,011.90
			5000	CSC Testing	4,940.00
	2752	Non-Real Time	4000	Implementation	1,086.80
			5000	CSC Testing	4,633.20
	2751	In House Dev/Test			7,651.20
	2753	On Site Install & Integ			154.00
	2754	Final Test			677.80
	2757	Config Audit			48.00
	2780	Reviews & Conf			102.00
	2563	Phase III Eng Data			1,102.00
	2779	Quality Assurance			430.00
	2777	S/W Config Mgmt			350.00
				Total Hrs Phase III	22,186.90
				Total All Phases	37,626.90

d. **Average No. of Hours per Staff Month:** 151 (default = 152 hours).

APPENDIX B

INSTRUCTIONS FOR

SOFTWARE PROJECT DATA COLLECTION FORMS

PROJECT QUESTIONNAIRE

GENERAL INFORMATION

1. **Your Name and Date**

Identify the person completing the questionnaire and the date that the form is being completed.

2. **Title and Phone**

Enter the title of the person completing the questionnaire and the number at which they can be reached.

3. **Organization and Address**

Identify the company or organization of the person completing this form.

4. **Name of Project**

Enter the name of the project for which the software is being developed.

5. **Contract Number**

If the software is developed under government contract, enter the prime contract number.

6. **Customer Name**

Enter the name of the organization for whom the software is being developed.

7. **Project Overview Description**

Describe the overall mission or purpose of the system for which the software is being developed.

8. **Developer Contact and Phone**

Enter a point of contact of the company or organization which is actually performing tne software development and the number at which they can be reached.

9. **Customer Contact and Phone**

Enter a customer point of contact and the number at which they can be reached.

10. **Current Status**

Enter whether the project is completed or ongoing. If ongoing, indicate the most recently completed project milestone.

PROJECT QUESTIONNAIRE

PRODUCT DESCRIPTION

1. **System/Software Characteristics**

   a. **Operating Environment:**

   (SASET: Class of Software)
   Select the operating environment of the target system.

   b. **Applications domain**

   (SoftCost-Ada: Type of Software)
   Select the appropriate software application domain for the project. The following types of systems can be designated:

   Automation - The software will be used in process control systems, such as those used for environmental control in a manufacturing plant.

   Avionics - The software will be used in avionics and other embedded systems, such as those used to control complex, real-time radars and guidance and control systems.

   Command & Control - The software will be used in command and control systems, such as air traffic control systems.

   Data Processing - The software will be used in traditional data processing systems, such as management information systems, payroll, accounting, time cards, etc.

   Environment/Tools - The software will be used in software development tool systems, such as compilers, CASE, and integrated software engineering environments.

   Scientific - The software will be used in scientific applications, such as seismic processing or weather mapping.

   Simulation - The software will be used in simulation systems, such as aircraft flight simulators.

   Telecommunications - The software will be used in telecommunications systems, such as digital switches or PABX's.

   Test - The software will be used in test systems, such as those used to monitor the performance application software.

   Other - Other types of applications not included in those listed above.

2. *Complexity*

   a. **Rate the difficulty of the processing logic**

   (SoftCost-Ada: Product Complexity)
   The following explanations are offered to assist with rating selections:

PROJECT QUESTIONNAIRE

PRODUCT DESCRIPTION

Strait line code, standard types *- The software will perform very basic functions using Ada's standard types. It will use basic math operations and will not use Ada's tasking conventions. An example of software with this amount of complexity is a screen generator or report writer.*

Simple functions, standard types *- The software will perform a basic set of functions using standard types, basic math operations, and no tasking. It may include some data manipulation routines and library calls. An example of software with this complexity level is a simple device driver or file management routine.*

Strait forward logic, generics and simple tasking *- The software will perform a set of functions using straightforward logic and I/O processing. It uses simple tasking primitives and generates/uses some generics. An example is scientific software used to compute the radius of an ellipsoid in three dimensions.*

Highly nested logic, numeric types, concurrent tasking *- The software will perform some real-time functions. It will be logically complex with complicated I/O structures and highly nested logic. It will generate and use packages and generics from a reuse library. It will also make use of Ada's numeric types and will handle multiple tasks executing concurrently. An example is an exception handler.*

Stochastic logic, unique type, rendezvous *- The software will perform real-time functions which have significant interface and interaction requirements. It will employ sophisticated math functions, user defined types, a reuse library and Ada's rendezvous facility for task synchronization. An example is a scheduler or simple control system.*

Dynamic resource allocation, unique types, rendezvous *- The software will perform real-time functions, like signal processing, which have extremely complex interfaces, control logic and time-dependent processing needs. It performs very difficult, unstructured numerical analysis functions, makes use of user defined types, incorporates very specialized libraries of package and generic units and contains very complicated exception handling provisions. Most military avionics and command and control systems fit this category.*

b. The complexity of this CSCI is best characterized by which of the following statements?:

(Ada COCOMO: Software Product Complexity)
Select the statement which best characterizes the complexity of your application.

c. Degree of Real-time

(SoftCost-Ada: Degree of Real-Time)
The following explanations are offered to assist with ratings selections:

Essentially batch response *- The software will perform in batch mode with no interactive or real-time response requirements.*

Interactive with limited Ada tasking *- The software will perform in an interactive mode, with a limited amount of Ada tasking.*

Interrupt driven with millisecond tasking *- The software will perform in a real-time mode, be interrupt driven and able to handle task communication in the millisecond time range.*

PROJECT QUESTIONNAIRE

PRODUCT DESCRIPTION

Concurrent tasking with millisecond rendezvous *- The software will perform in a real-time mode, support concurrent tasking and be able to support rendezvous which occur in the millisecond time range.*

Concurrent tasking with nanosecond rendezvous *- The software will perform in a real-time mode, support concurrent tasking and be able to support rendezvous which occur in the nanosecond time range.*

3. Reliability

a. Effect of a software failure

(Ada COCOMO: Required Software Reliability)
The following explanations are offered to assist with rating selections:

Inconvenience *- The effect of a software failure is simply the inconvenience incumbent on the developers to fix the fault. Typical examples are a demonstration prototype of a voice typewriter or an early feasibility-phase software simulation model.*

Easily-Recoverable Loss *- The effect of a software failure is a low level, easily-recoverable loss to users. Typical examples are a long-range planning model or a climate forecasting model.*

Moderate loss *- The effect of a software failure is a moderate loss to users, but a situation from which one can recover without extreme penalty. Typical examples are management information systems or inventory control systems.*

Major financial loss *- The effect of a software failure can be a major financial loss or a massive human inconvenience. Typical examples are banking systems and electric power distribution systems.*

Loss of human life *- The effect of a software failure can be the loss of human of life. Examples are military command and control systems or nuclear reactor control systems.*

4. Interfaces

a. Man-machine Interaction

(SASET: Man Interaction)
Address the level of man interaction inherent in the system. The more extensive man interactive systems are generally more expensive and take longer to build due to special input and error detection and correction functions that are needed.

b. Software Interface Complexity

(SASET: Software Interfaces)
Enter the number of software systems and peripheral communications equipment with various protocols and baud rates that this software system will interface with?

PROJECT QUESTIONNAIRE

PRODUCT DESCRIPTION

5. Software Testability

(SASET: Software Testability)
Systems possessing performance operations that are difficult to test are generally more expensive and take longer to build due to added complexity of the testing phase.

Very difficult software system to test *- long running programs with extensive logical paths to check.*

Difficult software system to test *- long running programs with many logical paths to check*

Time intensive program *- requires extensive testing but will not have a high degree of difficulty*

Program is easy to test *- small number of items to test*

6. Reused Code

a. Select the intended use of the majority of the software packaged for reuse

(Ada COCOMO: Required Reusability)
The rating selected should reflect added design, documentation, and more extensive testing associated with developing reusable Ada components.

b. Reuse Costs

(SoftCost-Ada: Reuse Costs)
The following explanations are offered to assist with ratings selection:

No reuse library *- Neither a reuse library nor a set of technical guidelines have been established by the firm. The costs of establishing the reuse infrastructure will be borne by the project.*

Reuse library employed *- A reuse library has been established for managing reusable artifacts. The library is not well populated and technical guidelines for packaging, quality assurance, and configuration management of reusable components are under development. The costs associated with trial use and refinement of the infrastructure will be borne by the project.*

Reuse library being populated *- A reuse library has been established and is current being populated. Technical and managerial guidelines for reuse have been published. The costs associated with use of the infrastructure will be borne partially by the project and possibly a process group as the library is being populated.*

Reuse library being exploited *- A reuse library has been established and populated, and is being exploited on the project. Technical guidelines for reuse have been published and people have been trained in their use. The costs associated with use of the infrastructure will be borne by the project as will their proportionate share of the costs associated with operating the library.*

PROJECT QUESTIONNAIRE

DEVELOPMENT METHODOLOGY

1. **Milestones**

a. Schedule

(SASET: Schedule)
Enter the expected and actual dates for each milestone, or N/A if the milestone does not apply to this project. Several CSCIs may be involved and they do not necessarily need to adhere to the same schedule. If an expected date is an estimated date rather than a contract date, put an asterisk after that date. The format for software development schedule date is (Month/Year).

b. Percent of development schedule devoted for Preliminary Design phase

(Ada COCOMO Σ Factor: Risk Elimination By PDR)
(Ada COCOMO Σ Factor: Design Thoroughness By PDR)
Select the percentage that most closely approximates the percentage of time devoted to the prelimary design phase based on a total time starting with Software Specification Review and ending with Formal Qualification Review.

2. **Development Standards**

a. Check all types of standard used in this development

(SoftCost-Ada: Degree of Standardization)
The following explanations are offered to assist with rating selections:

None *- No software development standards are available or will be used on the project.*

Ada Programming Standards *- The project will use a set of Ada programming standards that apply primarily to the coding of Ada software.*

Commercial Life Cycle Standards *- The project will use commercially developed (IEEE Standards, etc.) or company developed and client approved standards that apply to the design, development, and documentation of the Ada software.*

Military Standards *- The project will used a set of military standards on the project. Military standards typically employed include DOD-STD-2167, DOD-STD-2167A and DOD-STD-2168.*

b. List the name(s) of these standard(s)

(SASET: Software Documentation)

c. Were these standards tailored specifically for use on this effort?

(SoftCost-Ada: Degree of Standardization)
(SASET: Software Documentation)

DEVELOPMENT METHODOLOGY

Untailored standards means that the project will be forced to design, develop, and document software by the book. No waivers or deviations to the standards will be allowed.

d. List the name(s) of the software documents required

(SASET: Software Documentation)

3. Risk Management

a. Number and criticality of risk items

(Ada COCOMO Σ Factor: Risk Elimination By PDR; not required by COSTMODL)

b. Risk Management Plan identifies all critical risk items, establishes milestones for resolving them by PDR

(Ada COCOMO Σ Factor: Risk Elimination By PDR)

c. Schedule, budget, and internal milestones through PDR compatible with Risk Management Plan

(Ada COCOMO Σ Factor: Risk Elimination By PDR)
(Ada COCOMO Σ Factor: Design Thoroughness By PDR)

d. Tool support available for resolving risk items

(Ada COCOMO Σ Factor: Risk Elimination By PDR)

4. Software Reviews

a. Select all informal reviews held on the software during this development

(SoftCost-Ada: Use of Peer Reviews)
The following explanations are offered to assist with rating selections:

Quality Inspections/Audits *- The project will have quality assurance independently inspect/audit the software designs and code to ensure that they meet standards.*

Design and code walkthroughs *- The project will have software team members review each others' designs and code using the concept of walkthroughs. Walkthroughs are informal meetings where team members review work and suggest ways to improve it. Walkthroughs are used to improve the quality of the product.*

Design and code inspections *- The project will have software team member review each others' designs and code using the concept of inspections. Inspections have the same objectives as walkthroughs, but tend to be more formal. They are moderated (often by quality assurance) and feed-forward and feed-back the*

DEVELOPMENT METHODOLOGY

results of the review so that their lessons learned can be propagated throughout the project.

b. Select all management reviews held on the software for this project

(SoftCost-Ada: Use of Peer Reviews)
Management Reviews *- Peer management reviews are employed to build the management team and to have them help each other to solve problems and to manage risk.*

5. System/Software Requirements

a. Select the option which corresponds to the level of definition and understanding of system requirements

(SASET: System Requirements)

b. Select the option which corresponds to the level of definition and understanding of software requirements

(SASET: Software Requirements)

c. How will overall technology changes impact the project?

(SASET: Technology Impacts)

d. Select the percentage of software requirements well established

(SoftCost-Ada: Requirements Volatility)
The following explanations are offered to assist with ratings selections:

Essentially no changes (>90%) *- The software requirements are well defined and will change very little during the course of the development. Requirements changes will be infrequent and under change control.*

Over 60% of requirements are well established *- More than 60% of the software requirements are well established and will change slightly during the course of development. The remaining requirements will be defined and placed under control by SSR. Requirements changes will be infrequent and under change control.*

Over 50% of requirements are well established *- Between 50% and 60% of the software requirements are well established and will change during the course of development. The remaining requirements will be defined and placed under change control by SSR. Requirements changes will be frequent, but under change control.*

Over 30% of requirements are well established *- Between 30% and 50% of the software requirements are well established and will change during the course of development. The remaining requirements will be defined and placed under change control between SSR and PDR. Requirements changes will occur* *frequently and will result in moderate to heavy rework. Change control will be implemented, but will be heavily taxed to keep up with the requirements changes.*

Less than 30% of requirements are well established *- Between 0% and 30% of the software requirements are well established and will change during the course of development. The remaining requirements will be defined and placed under change control by PDR. Changes to requirements will occur fairly frequently and will require extensive rework. Change control will be implemented, but will be taxed to keep up with the requirements changes. Some thrashing will occur as products have to be reworked to accommodate requirements growth.*

e. System requirements baselined, under rigorous change control

(Ada COCOMO Σ Factor: Requirements Volatility)

f. Level of uncertainty in key requirements areas: mission, user interface, hardware, other interfaces

(Ada COCOMO Σ Factor: Requirements Volatility)

g. Organizational track record in keeping requirements stable

(Ada COCOMO Σ Factor: Requirements Volatility)

h. Use of incremental development to stabilize requirements

(Ada COCOMO Σ Factor: Requirements Volatility)

i. System architecture modularized around major sources of change

(Ada COCOMO Σ Factor: Requirements Volatility)

j. Level of uncertainty in key architecture drivers: mission, user interface, hardware, COTS, technology, performance

(Ada COCOMO Σ Factor: Design Thoroughness By PDR)

6. Commercial off-the-shelf software (COTS)

a. Select the option which best describes the expected impact of integrating commercial off-the-shelf software into the system

(SASET: COTS Software)

7. Use of Software Tools

PROJECT QUESTIONNAIRE

DEVELOPMENT METHODOLOGY

a. **Specify the type of environment that will be used to develop the software**

(SASET: Software Development Tools)
(SoftCost-Ada: Use of Software Tools/Environment)
The following explanations are offered to assist with rating selections:

Basic Ada Language Tools *- The minimum set of Ada software development tools will be used by the project. These tools typically include a text editor, compiler, linker/loader, and debugger.*

MAPSE Plus Host Tools *- A Minimal Ada Program Support Environment (MAPSE) will be used by the project which as host tools but no back-end target tools (i.e., no cross development tools). A MAPSE integrates the following types of tools into a software development environment: command language interpreter, text editor, compiler, debugger, linker/loader, static analyzer, dynamic analyzer, pretty-printer, file manager, and library.*

MAPSE Plus Host/Target Tools *- A MAPSE will be used by the project which has both host and back-end target tools. In addition to a standard MAPSE, this type of Ada programming environment provides cross-development tools which allow software to be written on the host and downloaded to the target after debugging has taken place.*

APSE *- An Ada Programming Support Environment (APSE) is richer in tools than a MAPSE because it provides the following additional types of tools: documentation systems, configuration management systems, project management systems, upper CASE and lower CASE tools.*

Full, Integrated, Life Cycle APSE *- An APSE which provides an integrated set of tools will be used on the project. This type of environment provides tools which are integrated with each other and the methods which they automate to provide a seamless system which can be used to support software development from start to finish.*

b. **Specify the type of tools that will be used to develop the software**

(Ada COCOMO: Use of Software Tools)
The following explanations are offered to assist with rating selections:

Basic microprocessor tools *- Assembler, Basic linker, Basic monitor, Batch debug aids*

Basic minicomputer tools *- HOL compiler, Macro assembler, Simple overlay linker, Language independent monitor, Batch source editor, Basic library aids, Basic database aids*

Strong mini, Basic maxicomputer tools *- Real-time or timesharing operating system, Database management system, Extended overlay linker, Interactive debug aids, Simple programming support library, Interactive source editor*

Strong maxi, Stoneman MAPSE *- Virtual memory operating system, Database design aid, Simple program design language, Performance measurement and analysis aids, programming support library with basic CM aids, Set-use analyzer, Program flow and test case analyzer, Basic text editor and manager*

DEVELOPMENT METHODOLOGY

Advanced maxi, Stoneman APSE *- Full programming support library with CM aids, Full integrated documentation system, Project control system, requirements specification language and analyzer, Extended design tools, Automated verification system, Special-purpose tools: Crosscompilers, instruction set simulators, display formatters, communications processing tools, data entry control tools, conversion aids, etc.*

c. Tool support for developing and verifying Ada package specs

(Ada COCOMO Σ Factor: Design Thoroughness By PDR)

8. Use of Modern Programming Practices

a. Degree to which modern programming practices are used in developing software

(Ada COCOMO: Use of Modern Programming Practices)
(Ada COCOMO Σ Factor for Maintenance Model: Use of MPPs)
The specific practices included here are:

1. ***Top Down Requirements Analysis and Design.*** *Developing the software requirements and design as a sequence of hierarchical elaborations of the users' information processing needs and objectives. This practice is extended to include the appropriate use of incremental development, prototyping, and anticipatory documentation.*

2. ***Structured Design Notation.*** *Use of a modular, hierarchical design notation (program design language, structure charts, HIPO) consistent with the structured code constructs in item 5.*

3. ***Top Down Incremental Development.*** *Performing detailed design, code, and integration a sequence of hierarchical elaborations of the software structure.*

4. ***Design and Code Walkthroughs or Inspections.*** *Performing preplanned peer reviews of the detailed design and of the code of each software unit.*

5. ***Structured Code.*** *Use of modular, hierarchical control structures based on a small number of elementary control structures, each having only one flow of control in and out.*

6. ***Program Librarian.*** *A project participant responsible for operating an organized repository and control system for software components.*

b. Ada Development Methodology

(SoftCost-Ada: Use of Modern Software Methods)
The following explanations are offered to assist with ratings selections:

Structured Programming *- The project will use traditional structure methods to analyze, design, develop, and test the software (e.g., including structured analysis, structured design, top-down development, program libraries, etc.)*

DEVELOPMENT METHODOLOGY

Object Oriented Design Plus Structured Programming *- A project will use a combination of structured programming techniques and Object Oriented Design (OOD). OOD is a techniques whereby a system is partitioned into object, not functions.*

Ada Packaging Methods *- The project will use Ada packaging methods based on object-oriented techniques in which an object and its operations are located within a single package.*

Integrated life-cycle methodology which exploits Ada reusability concepts *- The project will use an integrated set of object-oriented methods which enable its users to package the software to take full use of Ada's structural, behavioral, performance, tasking, and reuse features.*

c. Maintenance Conformance to the Ada Process Model

(Ada COCOMO Σ Factor for Maintenance Model: Conformance)

PROJECT QUESTIONNAIRE

SOFTWARE SIZE

1. **Size Estimates**

   a. **Number of CSCIs:**

(SASET: Number of CPCIs)
Computer software (program) configuration items (CSCIs) are identified early in the requirements phase along with hardware configuration items (HWCIs). Software CSCIs are complete, stand-alone, well-defined, and completely testable items.

   b. **Identify counting convention which is used to provide requested sizing information in (c).**

Counting conventions are recommended for the following models:

Model	*Convention*
Ada COCOMO	*Body semicolons*
SoftCost-Ada	*Terminal semicolons*
SASET	*Terminal semicolons*

Definitions for an Ada source line of code are as follows:

***Physical lines** - Any carriage return or line feed including comments and blank lines. Reusable code is counted the first time it is instantiated.*

***Non-Comment, Non-Blank Lines** - Physical lines excluding comments and blank lines.*

***Terminal Semicolons** - A statement terminated by a semicolon, including data declarations, code used to instantiate a reusable component, and the reusable component itself (the first time it was instantiated). Comments, blank lines, and non-deliverable code are not included in the line count.*

***Essential Semicolons** - Terminal semicolons excluding those used in a data declaration or formal parameter lists.*

***Body Semicolons** - A statement terminated by a carriage return in the specification and a terminal semicolon in the body of an Ada program. Comments, blank lines, and non-deliverable code are not included in the line count.*

   c. **Enter the requested sizing information below, in thousands of lines of source code (KSLOCs).**

(SoftCost-Ada: Ada Usage Factor)
(SoftCost-Ada: New, Reused, Modified Ada Components)
(SoftCost-Ada: New, Reused, Modified Other Components)
(SASET: Primary Software Language)
(SASET: Programing Language)
(SASET: Direct Input for SLOC)
Specify deliverable program or CSC, Number of lines of code for each code condition, language, and software type.

PROJECT QUESTIONNAIRE

SOFTWARE SIZE

Software Type can be system, application, support, or security. The code conditions (new, modified, rehosted) are briefly defines as follows:

***New code** - This constitutes software code that is to be developed from scratch. Software requirements must be determined, a design established, the design must be coded and units tested, and the system integration must be tested.*

***Modified code** - This constitutes software code which has some development already complete and which can be utilized in the software program under consideration. Inherited or legacy software are terms often used for modified code. Generally, modified code at the very least needs to be retested and often some redesign and recoding efforts are required.*

***Rehosted code** - This consists of completed and tested software code which is to be transferred from one computer system to another. The computer systems are functionally different to the point of requiring some changes to existing code. Generally, the code requires no requirement definition, little or no design definition, and partial testing.*

d. Reused software

(SoftCost-Ada: Reuse Benefits)
Specify the amount of software (design, code, tests, etc.) that will be incorporated into the project currently being developed.

e. Number of delivered source instructions adapted from existing software to form the new product

% of adapted software's design modified in order to adapt it to new environment

% of adapted software's code modified in order to adapt it to new environment

% of effort required to integrate the adapted software into the new product and to test the resulting product as compared to the normal amount of integration and test effort for software of comparable size

(Ada COCOMO: Adapted Code)

2. **Database Size**

a. Database size

(Ada COCOMO: Database Size)
(SoftCost-Ada: Database Size)
Identify the relative size of the database represented as a percentage of the total program size. For example, if the program is 10,000 source lines of code (delivered source instructions) and the database is less than 1,000 bytes, then the database is less than 10% of the program size. The percentage may exceed 100%.

PROJECT QUESTIONNAIRE

PROJECT STAFFING

1. **Staff Size/Availability**

a. **Staff availability**

(SoftCost-Ada: Staff Resource Availability)
Identify the availability of staff required by the project that are available when needed to perform software development activities. The nominal rating for this parameter is between 30% and 50% availability.

b. **Percent of required top software architects available to project**

(Ada COCOMO Σ Factor: Risk Elimination By PDR)
(Ada COCOMO Σ Factor: Design Thoroughness BY PDR)

c. **Difficulty of staffing due to special training and clearances**

(SASET: Personnel Resources)

2. **Staff Skill/Experience**

a. **Skill Level of Analysts**

(Ada COCOMO: Analyst Capability)
(SoftCost-Ada: Analyst Capability)
Identify the relative capability of the analysts that will be used on the project. For example, a rating of the bottom 15th percentile means that the analysts assigned to this project are, on average, ranked in the 15th percentile of all analysts (i.e., 85% of all analysts are better qualified). The major attributes to be considered in the rating are:

- *Analysis ability*
  Efficiency and thoroughness
  Ability to communicate and cooperate.

These attributes should weight equally. The evaluation should not answer the level of experience of the analysts. The evaluation should be based on the capability of the analysts as a team rather than as individuals.

b. **Skill Level of Programmers**

(Ada COCOMO: Programmer Capability)
Identify the relative capability of the programmers that will be used on the project. For example, a rating of the bottom 15th percentile means that the programmers assigned to this project are, on average, ranked in the 15th percentile of all programmers (i.e., 85% of all programmers are better qualified). The major attributes to be considered in the rating are:

- *Programmer ability*

PROJECT STAFFING

- *Efficiency and thoroughness*
- *Ability to communicate and cooperate.*

These attributes should weight equally. The evaluation *<u>should not</u>* *answer the level of experience of the programmers. The evaluation should be based on the capability of the programmers as a* *<u>team</u>* *rather than as individuals.*

c. Average experience with similar applications

(Ada COCOMO: Applications Experience)
(SoftCost-Ada: Applications Experience)
Identify the average experience the software team has had with applications of like type, size, and complexity. Experience is based on the average of the entire project team, not any one individual. For example on a team with 2 people: one person has 10 years application experience and one has 2 years experience, then average = 6 years.

d. Average level of virtual machine experience of the <u>project team</u> developing the software module

(Ada COCOMO: Virtual Machine Experience)
For a given software system, the underlying virtual machine is the complex of hardware and software that the system calls upon to accomplish its tasks. For example:

- *If the subsystem to be developed is an operating system, the underlying virtual machine is the computer hardware*

- *If the subsystem to be developed is a database management system (DBMS), the underlying virtual machine generally consists of the computer hardware plus an operating system.*

Programming language is not considered part of the virtual machine.

e. Host Machine Expertise

(SASET: Hardware Experience)

f. Software Language and Operating System Expertise

(SASET: Software Experience)

g. Experience with chosen development methodology

(SoftCost-Ada: Ada Methodology Experience)
Identify the average experience the software team has had with the development methodology (i.e. object oriented development, structural model) which will be used on the project. Experience is based on the average of the entire team, not any one individual at the beginning of the project. The following explanations are offered to assist with ratings selections:

PROJECT QUESTIONNAIRE

PROJECT STAFFING

Just starting (less than 3 months) *- The team will have no practical experience using new Ada methods and will be unfamiliar with Ada concepts. They may be undergoing training.*

Limited experience (3 - 6 months) *- The team may be familiar with methods, but unable to take advantage of them because they have less than 6 months of experience using them.*

Experienced (6 - 12 months) *- The team will be experienced with the language but will be unable to use its underlying software engineering concepts because their experience of less than a year is still too limiting.*

Extensive Experience (1 - 2 years) *- The team will be experienced with methods and will be able to use most of their capabilities to perform their work. Underlying principles are exploited.*

Ada Pro (over 2 years) *- The team will be staffed with Ada professionals who have over two years of experience which qualifies them to take advantage of the language to its utmost.*

h. Experience with Ada Process Model

(Ada COCOMO: Experience with Ada Process Model)
The Ada Process Model is a process model for software development to reduce project inefficiency when large numbers of project personnel are working in parallel on tasks which are closely intertwined and incompletely defined. Features of the Ada Process Model include the following:

- *Produce compilable, compiler-checked Ada package specifications (and body outlines), expressed in a well-defined Ada Program Design Language (PDL), for all top-level and critical lower-level Ada components by the project's or increments PDR.*
- *Identify and eliminate all major risk items by PDR.*
- *Use a phased incremental development approach with the requirements for each increment (called a "build") stabilized by the build's PDR.*
- *Use small up-front engineering and design teams, with expertise in software architecture, Ada, and the applications domain.*
- *Use a project risk management plan to determine the approach for eliminating risk items by PDR, and also to determine the sequence of development increments.*
- *Use intermediate technical walkthroughs in the early requirements and design phases.*
- *Use individual detailed design walkthroughs for each component and technical walkthrough for each build instead of a massive CDR.*
- *Use continuous integration via compiler checking of Ada package specifications and incremental demonstration, rather than beginning integration at the end of unit test.*
- *Use bottom-up requirements verification via unit standalone tests, build integration tests, and*

engineering string tests.

- *Provide well-commented Ada code and big-picture design information instead of massive as-built Software Design Documents, which rapidly get out of date and loose their maintenance value.*

- *Use a set of consistent metrics tightly coupled to the project's Software Development Plan and its build definitions to provide visibility into the code development process.*

i. Project team's equivalent duration of experience (at the beginning on the project/build) with the programming language to be used

(Ada COCOMO: Programming Language Experience)
(SoftCost-Ada: Ada Language Experience)
Identify the average experience the software team has had with the programming language. Experience is based on the average of the entire team, not any one individual at the beginning of the project.

j. Number of Ada projects completed by team members

(SoftCost-Ada: Number of Ada Projects Completed)
Specify the average number of Ada software development projects completed by the development team. An Ada project is defined as the delivery of a product, packaged and prepared using Ada concepts (i.e., an incremental build, a prototype, a software delivery, etc.). The average is based upon the entire team including designers and senior analysts. If you are estimating an incremental development, reflect the number of completed builds. For example, if this was the third build of your first Ada project, then you would rate this factor as a 2 if none of your people have had Ada experience on previous projects.

k. Ada environment experience

(SoftCost-Ada: Ada Environment Experience)
Identify the average experience the analysts who are part of the team have had with the tools, equipment, and facilities that are part of the development environment to perform similar software development tasks. Base the number on the average of the entire team, not any one individual.

l. Level of product familiarity of the development team

(SASET: Development Team)

3. Teamwork Capability

a. Select the type of team used for software development

(SoftCost-Ada: Team Capability)
Identify the types of teams which will be used on the project. The following explanations are offered to assist with rating selections:

PROJECT QUESTIONNAIRE

PROJECT STAFFING

Design Teams *- The software will be designed by a team of analysts who may not be involved in the implementation. Personnel outside of the project may be called in to work specific problems and to collaborate in the design.*

Programming Teams *- The software will be designed, developed, and tested by a team of analysts who are involved in the project from its start to finish. Team reviews and approaches to development will be used as the team leader keeps control of the software development activities.*

Participatory Teams *- The software will be designed, developed, and tested by a team of analysts who use the consensus process to arrive at both technical and managerial decisions.*

Interdisciplinary Teams *- Both hardware and software personnel are collocated and work as a single team to solve their individual and interdisciplinary problems on the project using the consensus process.*

PROJECT QUESTIONNAIRE

COMPUTER SYSTEM

1. Development Environment

a. Number of different types of workstations

(SASET: Workstation Types)
A workstation is considered unique if it requires different screen clearing and set-up operations. This value can range from 0 to 100.

b. Rate the virtual machine volatility of the development system, based on frequency of major/minor changes

(Ada COCOMO: Virtual Machine Volatility - Host)
For a given software system, the underlying virtual machine is the complex of hardware and software that the system calls upon to accomplish its tasks. For example:

- *If the subsystem to be developed is an operating system, the underlying virtual machine is the computer hardware.*
- *If the subsystem to be developed is a database management system (DBMS), the underlying virtual machine generally consists of the computer hardware plus an operating system.*

Ratings which are defined in terms of relative frequency of major and minor changes are defined as follows:

- *Major change: significantly effects roughly 10% of routines under development.*
- *Minor change: significantly effects roughly 1% of routines under development.*

c. Select the following option that best assesses the embedded features of the development system

(SASET: Embedded Development System)

d. Rate the software tool/environment stability of the development system

(SoftCost-Ada: Software Tool/Environment Stability)
Identifies how stable the tools that will be used on the project are and how often changes in the environment will be processed. The following explanations are offered to assist with ratings selections:

Buggy Compiler *- The project will use a compiler which has not been thoroughly debugged and does not fully implement the full set of requirements set forth in the Ada Language Specification.*

Stable Compiler, Unstable Environment *- The project will use a compiler which has been fully debugged, but does not fully implement all of the requirements of the Ada Language Specification. The tool environment is unstable with changes occurring monthly. New tools or versions of old tools are being inserted into the environment weekly.*

Stable Compiler, Mature Environment *- The project will use a compiler which has been fully debugged and* *implements all of the requirements of the Ada Language Specification. The APSE is maturing with changes occurring quarterly. New tools or versions of old tools are being inserted into the environment monthly.*

Stable Compiler, Stable Environment *- The project will use a compiler which has been fully debugged and implements all of the requirements of the Ada Language Specification. The APSE is stable with changes occurring once a quarter. New tools or versions of old tools are being inserted into the environment quarterly.*

Stable Environment *- The project will use a compiler which has been fully debugged, implements all of the requirements of the Ada Language Specification, and is capable of supporting efficient tasking. The APSE is very stable with changes occurring semi-annually. New tools are being inserted into the environment quarterly.*

Mature, Stable Environment *- The project will use a compiler which has been fully debugged, implements all of the requirements of the Ada Language Specification, supports tasking and has been validated. The APSE is very stable with changes occurring semi-annually. New tools are being inserted into the environment semi-annually with a minimum of disruption.*

e. Address the difference between the development hardware system and the host system

(SASET: Development Versus Host System)

2. Target Computer Configuration

a. Rate the virtual machine volatility of the target system, based on number of major/minor changes

(Ada COCOMO: Virtual Machine Volatility-Target)
For a given software system, the underlying virtual machine is the complex of hardware and software that the system calls upon to accomplish its tasks. For example:

- *If the subsystem to be developed is an operating system, the underlying virtual machine is the computer hardware.*
- *If the subsystem to be developed is a database management system (DBMS), the underlying virtual machine generally consists of the computer hardware plus an operating system.*

Ratings which are defined in terms of relative frequency of major and minor changes are defined as follows:

- *Major change: significantly effects roughly 10% of routines under development.*
- *Minor change: significantly effects roughly 1% of routines under development.*

b. Identify the system architecture

(SoftCost-Ada: System Architecture)

PROJECT QUESTIONNAIRE

COMPUTER SYSTEM

(SASET: Hardware System Type)
Specify the architecture of the target computer system. If the target computer system will use multiple processors, specify the number of processors.

Centralized *- The target computer system will use a single processor.*

Tightly-Coupled *- The target computer system will use multiple processors which are very tightly-coupled, typically sharing a common pool of memory.*

Loosely-Coupled *- The target computer system will use multiple processors which are connected in a loosely-coupled manner with each processor typically having its own memory resources.*

Federated *- The target computer system will use multiple functional processors which communicate via either a common system-level bus or a communications channel. The key aspect of this type of architecture is the term "functional processors". Each processor is dedicated to performing a specific function and passes control and data information across the bus or communications channel to the other processors.*

Distributed (centralized database) *- The target computer system will use multiple, distributed computers, sharing a common database, with the software distributed across these computers.*

Distributed (distributed database) *- The target computer system will use multiple, distributed computers, with the software and the database(s) distributed across these computers.*

3. Performance Requirements

a. Main Storage Constraint

(Ada COCOMO: Main Storage Constraint)
(SASET: Percent of Core Utilized)
Main storage refers to direct random access storage such as core, integrated-circuit, or plated-wire storage; it excludes such devices as drums, disks, tapes, or bubble storage. Select the percentage which best reflects the percentage of main storage expected to be used by the subsystem and any other subsystems consuming the main storage resources.

b. Overall Hardware Constraints. *Overall hardware refers to processor memory, I/O capacity, and throughput (i.e. CPU speed) available within the target computer system.*

(SASET - Hardware Constraints)
(SoftCost-Ada: Degree of Optimization)
Specify how much optimization must be performed to make the software run within the resource constraints of the target computer system. The following explanations are offered to assist with rating selections:

Less than 50% of Available Resources Used *- The target computer system has more than enough resources available. The developer need no optimize to fit the software into memory or to execute it within required time.*

COMPUTER SYSTEM

A maximum of 75% of Available Resources are Used *- When completed, the software will use no more than 75% of the available computer resources. The developers need to perform only a minimal amount of optimization to cause the software to run efficiently on the computer.*

A maximum of 85% of Available Resources are Used *- When completed, the software will use no more than 85% of the available computer resources. Resource restrictions will require the developers to perform some optimization to tailor the software to memory or realize time restrictions.*

A maximum of 95% of Available Resources are Used *- The software, when complete, will use no more than 95% of the available computer resources. This would be the case when the target computer system has severe resource restrictions which require the developers to use a variety of optimization techniques to ensure the software will run on the target machines.*

Close to 100% of Available Resources are Used *- The software, when completed, may exceed the available computer resources. This represents an extreme case when the target processor has a fixed amount of resources. The developers will be required to use a variety of optimization techniques, such as overlays, to ensure that the software will run within these constraints.*

c. **Execution Time Constraints.** *Select the percentage which best reflects the percentage of available execution time expected to be used by the subsystem and any other subsystems consuming the execution time resource.*

(Ada COCOMO: Execution Time Constraint)

d. **Select the criteria which reflects the performance constraints of the software system**

(SASET: Timing and Criticality)

4. Microprocessor Code

a. **Percentage of software functions that are to be implemented in firmware**

(SASET: Percentage of Microprocessor Code)
Percentage is with respect to the total software job. Microprocessor code may be hosted on a chip such as: ROM, PROM, EPROM, or any other hardware used for storing executable microprocessor instructions. The added complexity of downloading ("burning") and testing the microprocessor software increases the development complexity.

DEVELOPMENT ENVIRONMENT

1. Project Organization

a. Number of organizations within the company significantly involved during the software development

(SoftCost-Ada: Number of Organizations)
Specify the number of organizations directly involved in the software development effort. The following lists examples of typical organizations included within this count:

Software Development
Software Configuration Management
Software Quality Assurance
Software Project Management
Software Test (if independent)
Project Management
Project-level Configuration Management
Project-level Quality Assurance
System Engineering
System Test (if independent)
Independent Verification and Validation (IV&V)

Subcontractors and co-contractors should each be counted as one separate organization.

b. Scope of Support

(SoftCost-Ada: Scope of Support)
The following explanations are offered to assist with ratings selection.

***Liaison support** - The software organization will occasionally be called upon to provide limited support to other project organizations (e.g., system test, project management, etc.). Support is provided primarily in a review and working group capacity.*

***Extensive support to system test** - The software organization will provide support to the system test organization during the conduct of system-level testing (i.e., hardware/software integration and acceptance testing). Support includes participation in planning, executing, and documenting system-level tests.*

***Extensive support to system engineering & system test** - The software organization will provide extensive support to many organizations involved in the project (e.g., members of system-level design and test teams, developers if ICD's, participants in system-level review, etc.). In addition, Cost Schedule Control Systems Criteria (C/SCSC) or Cost Schedule Status Report (CSSR) reports will be filed per customer requirements. Such requirements generate a great deal of paperwork, in that, earned value must be computed, variances tabulated, and technical performance documented on a periodic basis.*

c. Organizational Interface Complexity

(SoftCost-Ada: Organizational Interface Complexity)

PROJECT QUESTIONNAIRE

DEVELOPMENT ENVIRONMENT

Describe the interface complexity between organizations involved in the software development effort. The following explanations are offered to assist with rating selections:

Single Interface With Collocated Customer *- The software will be developed by an organization which directly interfaces with only one other organization. The customer is located in the same facility as the developing organization.*

Single Interface With Single Customer *- The software will be developed by an organization which directly interfaces with only a single, remote customer through the software project manager.*

Multiple Internal and Single External *- The software will be developed by an organization which interfaces with other organizations within the same company (i.e., quality assurance, etc.) and the customer through the software project manager.*

Multiple Internal and Single External Interfaces *- The software will be developed by an organization which interfaces with other organizations within the same company and multiple customers through different personnel in the project management organization.*

Multiple Geographically Distributed Interfaces *- The software will be developed by an organization which is geographically distributed and interfaces with other geographically dispersed organizations, customers, co-contractors, and subcontractors through different personnel in the project management and marketing organizations.*

d. Number of locations at which software is developed

(SASET: Development Locations)
Enter a value between 1 and 100.

e. Number of customer locations

(SASET: Customer Locations)
(SASET: Information Travel Requirements)

2. Computer Resources

a. Characterize the development facilities and the perceived availability of the hardware (terminals and computers)

(SASET: Development Facilities)

b. Computer resource availability

(SoftCost-Ada: Computer Resource Availability)
Identify how available computer equipment and facilities are for the software development effort. The following explanations are offered to assist with rating selections:

DEVELOPMENT ENVIRONMENT

Extreme Equipment and Facility Limitations *- Little if any of the computer resources required will be available when needed by the project. Access to machines is difficult and machine time is limited.*

Computer Shared and Remotely Accessible *- The computer resources used for software development will be located at some remove site and will be shared by multiple projects, thereby creating conflict over access.*

Interactive Access to Dedicated Resources *- The computer resources used for software development will be dedicated to the project and will provide users with interactive access via terminals or workstations.*

Dedicated Facilities With Multiple LAN-servers *- The computer resources used for software development will be dedicated to the project and will provide users with convenient access to a variety of machines, file servers, and workstations via a Local Area Network.*

Software Factory *- The computer resources used for software development will be dedicated to the project, convenient to use, and provide access to an ergonomically-designed, attractive Software Factory which uses a variety of machines, workstations, and specialized Ada facilities to perform needed tasks.*

c. Select the average time required to submit a job to be run until the results are back in the developer's hand

(Ada COCOMO: Computer Turnaround Time)

3. Security and Privacy Restrictions

a. Classified Application

(Ada COCOMO: Classified Security Application)

b. Security Requirements

(SoftCost-Ada: Security Requirement)
Specify the level of security requirements imposed on the software development effort. The following explanations are offered to assist with rating selections:

Database Integrity *- The only security requirements imposed on the project will be those normally imposed to maintain database integrity and information privacy.*

Physical Security *- Security on the project will be handled via physical safeguards which include guards and intrusion alarm systems.*

Demonstrably Correct Trusted System *- B Level trusted system requirements and physical security safeguards will be required on the project. Access controls must be demonstrated and certified by independent parties.*

Verifiably Correct Trusted System *- A Level trusted system requirements and physical security safeguards will be required on the project. Security controls must be verified using sophisticated proof of correction techniques.*

DEVELOPMENT ENVIRONMENT

c. Internal Computer System Security Safeguards

(SASET: Software Security)
Identify the software security controls that are designed to provide internal computer security safeguards.

PROJECT QUESTIONNAIRE

RESOURCE ALLOCATION

1. <u>Effort</u>

   a. **Total Staff**

   Enter total staff months of effort end-to-end.

   b. **Minimum Staff Size**

   c. **For each software activity, please provide the total effort, by phase, in staff-months it took to complete**

   d. **Average No. of Hours per Staff Month**

   Default = 152 hours/month.

(This Page is Intentionally Left Blank)

APPENDIX C

STATEMENT PROFILER DEFINITIONS

Body Semicolons: A statement terminated by a carriage return in the specification and a terminal semicolon in the body of an Ada program, including data declarations and code used to instantiate a reusable component itself the first time it was instantiated. Comments, blank lines, and non-deliverable code are not included in the line count.

Data Manipulation: Program text containing any of the following keywords: PUT, PUT_LINE, GET, GET_LINE, READ, WRITE

Data Typing: Program text containing any of the following keywords: TYPE, SUBTYPE

Essential Semicolons: Terminal semicolons excluding those used in data declarations or formal parameter lists.

Exception: An exception is an error situation that may arise during program execution. To raise an exception is to abandon normal program execution to signal that an error has taken place. An exception handler is a portion of the program text specifying a response to the exception. Execution of such a program text is called handling the exception.

Generic: A generic is a template for a set of subprograms or for a set of packages. A subprogram or package created using the template is called an instance of the generic unit. A generic unit is one of the kinds of program unit.

Logical: Program text containing any of the following keywords: IF, CASE, LOOP, EXIT, ELSE, ELSEIF, WHEN, GOTO.

Objects: Number of variables and constants.

Physical Lines: Any carriage return or line feed including comments and blank lines. Reusable code is counted the first time it is instantiated.

Program Units: Number of subprograms, packages, generics, and tasks.

Mathematical: Program text containing any of the following keywords: +, -, *, /, MOD, REM, **, ABS

Task: A task is a program unit which operates in parallel with other parts of the program.

Tasking: Program text containing any of the following keywords: SELECT, ACCEPT, ABORT, TERMINATE

Terminal Semicolons: A statement terminated by a semicolon, including data declarations, code used to instantiate a reusable component and the reusable component itself the first time it was instantiated. When multiple semicolons are user within a declaration statement, the terminating semicolon is used to define the termination of a source line of code. For example, a package specification which included a statement that spans ten lines and is terminated by a single semicolon would count as one ASLOC. Comment, blank lines, and non-deliverable code are not included in the line count.

(This Page is Intentionally Left Blank)

APPENDIX D

DERIVATION OF FUNCTION POINT COUNT FOR THE REAL-TIME CSCI OF THE UH-1 FLIGHT SIMULATOR

This attachment contains a description of how function point parameters were counted for the Real-Time CSCI of the UH-1 helicopter flight simulator, excluding diagnostics and other support software. Counting conventions are presented by parameter type:

- External Inputs
- External Outputs
- Logical Internal Files
- External Inquiries
- External Interfaces.

There are ambiguities with regard to the function point analysis for training devices. Our resolution and interpretation of the guidelines is presented here. For each parameter type the description of how counts were derived contains the following information:

- **Key points** - a summary of the basic parameter definition with emphasis of certain key factors.

- **Potential types within the UH-1 FS** - situations in which elements were counted as this parameter type.

- **Description** - an annotated listing of each parameter that was counted and the complexity level that was assigned.

- **Total number of element types** - total count of elements for the specified parameter.

EXTERNAL INPUTS

Key Points:

- User data or user control information that enters the external boundary of the application
- It must change something inside the system
- It is unique if it has a different format or requires different processing logic

Potential Types Within the UH-1 FS

Trainee Station:

- Cockpit Controls and Panels

Instructor Operator Station:

- Instructor Panels
- Initial Conditions
- Malfunctions Initiated by the Instructor

Description	**Complexity**

Trainee Station Inputs:

Cockpit Instrument Panel (pp. 82,83 SRS, Vol 1)

Note: Fire Detector Test Switch and Fuel Gauge Test Switch were counted as inquiries.

1.	Pressure Altimeter a. Barometric Pressure	Low
2.	Marker Beacon a. Power Switch b. Sensing Switch c. Volume Control	Low
3.	Course Indicator a. Course Set Knob	Low
4.	Radio Magnetic Indicator a. Set Heading Control b. ADF/VOR No. 1 Bearing Pointer Control c. Compass Slaving Switch	Low

Engine Panel

1.	Low RPM Audio On/Off	Low
2.	Fuel Main On/Off	Low
3.	Int Aux Fuel Left/Off	Low

4.	Int Aux Fuel Right/Off	Low
5.	De-Ice On/Off	Low
6.	Governor Auto/Emer	Low

<u>Chip Detector Panel</u>

Note: Chip Detect Transmission/Tail Rotor Switch was counted as an inquirey.

1.	Force Trim On/Off	Low
2.	Hydraulic Control On/Off	Low

<u>Lighting Panels</u>

1. Instrument Lighting Panel — Low

   a. Instrument Console Lighting Control
   b. Instrument Pedestal Lighting Control
   c. Instrument Secondary Lighting Control
   d. Instrument Engine Lighting Control
   e. Pilot Lighting Control
   f. Co-Pilot Lighting Control

2. Dome Lights Panel — Low

   a. Dome Lights White/Off/Red Switch
   b. Pitot Heater On/Off Switch

3. Exterior Lights Panel — Low

   a. Exterior Lights Steady/Flash Switch
   b. Exterior Lights Dim/Bright Switch
   c. Anti-Collision Lights On/Off Switch

<u>DC and AC Power Panels</u>

1. DC Power Panel — Low

   Note: DC Voltmeter Selector Switch was counted as an inquirey.

   a. Main Generator Reset/On/Off Switch
   b. Battery On/Off Switch
   c. Starter Gen Switch
   d. DC Power Manual On/Normal On Switch

2. AC Power Panel — Low

   Note: AC Power Phase Selector was counted as an inquirey.

   a. Invertor Spare/Main Switch

3. DC Circuit Breaker Panels — Average

a. DC Circuit Breakers

4. AC Circuit Breaker Panel Low

a. AC Circuit Breakers

Radio Set Control Panels

1. FM Radio Set Control Panel Low

a. Mode Selector Switch
b. Megahertz Control
c. Kilohertz Control

2. UHF Radio Set Control Panel Low

a. Function Selector Switch
b. Mode Selector Switch
c. Preset Channel Control
d. Ten Megahertz Control
e. One Megahertz Control
f. Five-hundredths Megahertz Control

3. VHF Radio Set Control Panel Low

a. Power Switch
b. Megahertz Control
c. Kilohertz Control

4. VHF Navigation Set Control Panel Low

a. Power Switch
b. Megahertz Control
c. Kilohertz Control

5. ADF Control Panel Low

a. Mode Selector Switch
b. Band Selector Switch
c. Tune Control
d. Loop L-R Switch

6. TACAN Radio Set Control Panel Low

a. Function Selector Switch
b. Mode Selector Switch
c. Channel Select Control
d. Bit push button

7. Signal Distribution Panel Low

a. FM Receiver Switch
b. UHF Receiver Switch
c. VHF Receiver Switch
d. INT Switch
e. NAV Switch
f. Transmit-Interphone Selector Switch

Miscellaneous Panels

1.	Miscellaneous Control Panel	Low

a. Wiper Select Pilot/Co-Pilot Switch
b. Wiper Speed Select Switch

2.	Cabin Heating Panel	Low

a. Bleed Air Select Switch
b. Aft Outlet Select Switch

IFF Transponder Set Control Panel

Note: Master Control Off/Stby/Low/Norm/Emer Switch was counted as an inquirey.

Problem Control Panel

Note: INSTR CALL was counted as an inquiry.
Motion Controls are implemented completely in hardware and have no software impact.

3.	Turbulence Level (via Select Thumbwheel)	Low

Collective Pitch Control Lever

1.	Collective Pitch Lever Deflections	Average
2.	Throttle Position	Average
3.	Engine Idle Stop Release Switch	Low
4.	Starter-Ignition Switch	Low
5.	Governor RPM Switch	Low

Cyclic Control Stick

1.	Lateral and Longitudinal Cyclic Deflections	Average
2.	Force Trim Push Button Switch	Low

Pilot/Copilot Anti-Torque Pedals

1.	Directional Pedal Position	Average

Instructor Operator Station (IOS) Inputs:

Trainee Station Control Panel (p. 36, SRS, Vol 1)

Note: Motion Controls are implemented completely in hardware and have no software impact.
Hardcopy Controls are counted as inquiries
ACK STUD push button was counted as an inquirey.

1.	Mode Controls	Low	
	a. SEMI AUTO push button		
2.	Graphic Display Controls	Low	

Note: G TRK scaling, FULL SCALE AS, and FULL SCALE ALT were counted as inquiries.

a. G TRK ERASE push button
b. PLOT AREA RCL push button
c. AREA SEL thumbwheel switch
d. GCA COMM push button

3. Intercom Controls — Low

Note: Speaker and Volume Intercom controls are implemented in hardware and have no software impact.

a. HDST A push button
b. HDST B push button

4.	Playback Controls	Average	
	a. MIN SEL Thumbwheel		
	b. RESET push button		
	c. IN PROG push button		
	d. SLOW TIME push button		
	e. PAUSE push button		
5.	Malfunction Controls	Average	
	a. SEL Thumbwheel		
	b. INSR push button		
	c. INHB RMV push button		
	d. MALF push button (located on Problem Control Panel)		
	e. Select Thumbwheel (located on Problem Control Panel)		
6.	Crash Override Mode Controls	Low	
	a. CRASH OVRD push button		Trainee Station Control Panel
7.	Simulation Freeze/Continue Controls	Low	
	a. PROB FRZ push button		Trainee Station Control Panel
	b. FRZ push button		Problem Control Panel
	c. CONT push button		Problem Control Panel
8.	Reset Simulation to System Start-Up Conditions	Low	
	a. PROB RESET push button		Trainee Station Control Panel

	b. RESET push button		Problem Control Panel
9.	Automatic Copilot Mode Controls	Low	
	a. Enable AUTO COPILOT push button		Trainee Station Control Panel
	b. Disable AUTO COPILOT push button		Problem Control Panel

Auxiliary Information Display (AID) Control Panel

Note: The following controls were counted as inquiries:

- Display Area Select Controls
- Transfer Cockpit Area to Edit Area Controls

The following controls were counted as outputs:

- Display Select Controls
- Display/Edit Format Select Controls

1.	AUX MODE push button	Low
2.	Parameter Control	Average
	a. FLT PRMTR FRZ push button	
	b. FLT PRMTR RSTRE push button	

Communications Control Panel

1.	CM AUDIO NET push button	Low
2.	MON STUD HDST push button	Low
3.	ATC push button	Low
4.	Transmit push buttons	Low
	a. UHF push button	
	b. VHF FM push button	
	c. VHF NAV push button	
	d. ICS push button	

Instructor Initiated Malfunctions (described in MMR pp. 243-257)

Note: Similar malfunctions are grouped. Groupings are based on Object Interface Diagrams presented in the preliminary design.

1.	Malfunctions Affecting Tail Rotor Forces and Moments	Low
	a. Tail Rotor Gearbox (Group 3 Flight Malfunctions)	
2.	Malfunctions Affecting Flight Controls	Low
	Flight Malfunctions (Group 3)	
	a. Tail Rotor Loss	

b. Tail Rotor Thrust
c. Tail Rotor Fixed Pitch

3. Malfunctions Affecting Weight and Balance — Low

a. Tail Rotor Gearbox (Group 3 Flight Malfunctions)

4. Malfunctions Affecting Electrical Power System — High

Electrical System Malfunctions (Group 4)

a. Complete Electrical Failure
b. Main Generator
c. Standby Generator
d. Main Invertor
e. STBY Invertor

Indicator Circuit Breaker Malfunctions (Group 5)

a. Attitude Indicator Pilot #1 CB, OA
b. Attitude Indicator Pilot #2 CB, OC
c. Attitude Indicator Copilot #1 CB, OA
d. Attitude Indicator Copilot #2 CB, OC
e. Course Direction Indicator CB
f. Gyrocompass CB
g. Turn-and-Slip Indicator
h. Engine and Transmission Temp CB
i. Fuel Quantity Indicator CB
j. Fuel Pressure Indicator CB
k. Engine Oil Pressure Indicator CB
l. Transmission Oil Pressure Indicator CB
m. Torquemeter CB
n. Nonessential Bus VM CB

Navigation/Communication Circuit Breakers (Group 6)

a. VHF Transceiver CB
b. UHF Transceiver CB
c. FM Transceiver CB
d. Intercom - Pilot CB
e. Intercom - Copilot CB
f. IFF Transponder CB
g. ADF Compass CB
h. VHF Navigation Receiver CB
i. Marker Beacon CB

Illumination Circuit Breaker (Group 7)

a. Instrument Panel Lights CB
b. Utility Lights CB
c. Dome Lights CB
d. Caution Lights CB

e. Instrument Secondary Lights CB
f. Console and Pedestal Lights CB
g. Generator Reset CB
h. Invertor Control CB
i. Main Invertor Power CB
j. Spare Invertor Power CB
k. Alternating Current (AC) 115-Volt Relay CB
l. AC 115-Volt 28-Volt Transformer CB

Miscellaneous Circuit Breaker (Group 8)

a. Starter Relay CB
b. Ignition System CB
c. Governor Control CB
d. Engine Anti-Ice CB
e. Idle Stop Release CB
f. Fuel Valve CB
g. Right Fuel Boost Pump CB
h. Hydraulic Control CB
i. Force Trim System CB
j. Pitot Heater CB
k. RPM Limit Warning CB
l. Fire Detect CB

5. Malfunctions Affecting Caution Advisory Panel — Low

a. Master Caution Light (Group 4 Electrical System Malfunctions)

6. Malfunctions Affecting the Fuel System — Low

a. Fuel Quantity Indicator (Group 1 Indicator Malfunctions)
b. Fuel Pressure Indicator (Group 1 Indicator Malfunctions)
c. Left Fuel Boost Pump (Group 4 Electrical System Malfunctions)
d. Right Fuel Boost Pump (Group 4 Electrical System Malfunctions)

7. Malfunctions Affecting the UHF Radio — Low

a. UHF Transceiver (Group 4 Electrical System Malfunctions)

8. Malfunctions Affecting the FM Radio — Low

a. FM Transceiver (Group 4 Electrical System Malfunctions)

9. Malfunctions Affecting VHF Communications — Low

a. VHF Communications Transceiver (Group 4 Electrical System Malfunctions)

10. Malfunctions Affecting VHF Navigation — Low

a. Glide Slope Needle (Group 1 Indicator Malfunctions)
b. VHF Navigation Receiver (Group 4 Electrical System Malfunctions)
c. Marker Beacon Receiver (Group 4 Electrical System Malfunctions)

11. Malfunctions Affecting the ADF Radio — Low

a. LF-ADF Receiver (Group 4 Electrical System Malfunctions)

12. Malfunctions Affecting the Engine Power Train — Low

Engine/Transmission Malfunctions (Group 2)

a. No Start
b. Short Shaft Failure
c. Inlet Guide Vane Actuator

13. Malfunctions Affecting the Engine Gas Generator — Average

Engine/Transmission Malfunctions (Group 2)

a. Engine Fuel Pump
b. Flameout/total engine failure
c. Hot Start
d. Hung Start
e. Short Shaft Failure
f. Compressor Stall
g. Governor RPM Increase/Decrease Switch
h. Inlet Guide Vane Actuator
i. Governor, Low Side
j. Governor, High Side
k. Droop Compensator
l. Engine Tachometer Generator

14. Malfunctions Affecting Engine Lubrication — Low

a. Transmission Oil Loss - Abrupt (Group 2 Engine/Transmission Malfunctions)
b. Transmission Oil Loss - Gradual (Group 2 Engine/Transmission Malfunctions)
c. Engine Fire (Group 2 Engine/Transmission Malfunctions)
d. Engine Chip Detector Light (Group 4 Electrical System Malfunctions)
e. Chip Detector Light (Group 4 Electrical System Malfunctions)

15. Malfunctions Affecting the Control Loading System — Low

Flight Malfunctions (Group 3)

a. Tail Rotor Thrust
b. Lateral Cyclic
c. Lateral Cyclic Hardover
d. Longitudinal Cyclic
e. Longitudinal Cyclic Hardover
f. Total Hydraulics Failure

16. Malfunctions Affecting the Motion System — Low

Flight Malfunctions (Group 3)

a. Main Rotor Blade Track
b. Main Rotor Blade Balance
c. Tail Rotor Track

17. Malfunctions Affecting Instruments — Low

Indicator Malfunctions (Group 1)

a. Attitude Indicator - Pilot
b. Attitude Indicator - Copilot
c. Turn Needle
d. Gyromagnetic Compass Heading Indicator
e. Gyromagnetic Compass - Slave Failure
f. Pitot System Failure (icing)
g. N_1 Tachometer
h. Rotor Tachometer Generator
i. Torquemeter
j. Engine Oil Temperature Indicator
k. Engine Oil Pressure Indicator
l. Transmission Oil Temperature Indicator
m. Transmission Oil Pressure Indicator
n. Fuel Quantity Indicator
o. Fuel Pressure Indicator

<u>Initial Conditions and Flight Parameters</u>

1. Initial Conditions — Average

a. Altitude
b. Airspeed
c. Mag Heading
d. Roll
e. Pitch
f. Yaw
g. Vertical Velocity
h. Turn Rate
i. Torque Pressure
j. Rotor RPM
k. Latitude
l. Longitude
m. Fuel Weight
n. Center of Gravity
o. Gross Weight
p. Barometric Pressure
q. Outside Air Temperature
r. Wind Velocity
s. Wind Direction
t. Turbulence Level
u. Sound Level
v. Radio Static Level
w. Aux Power Unit
x. Fuel Burn Multiplier

2. Flight Parameters — Average

Total Number of UH-1 FS External Inputs

	Low	Average	High
Trainee Station Inputs:	32	5	0
Instructor Operator Station (IOS) Inputs:	28	6	1
TOTAL:	**60**	**11**	**1**

EXTERNAL OUTPUTS

Key Points:

- User data or user control information that leaves the external boundary of the application measured
- It is unique if it has a different format or requires different processing logic
- It does not include output response of an external inquiry

Potential Types Within the UH-1 FS

- Auxiliary Displays
- Maps
- Map Components
- Indicator Displays
- Station Identifiers

	Description	Complexity
	Map Displays - Problem Status Display Area (pp. 65-73)	
1.	Problem Status Information Display	High
	a. Training Mode Group	
	b. Air Traffic Control Group	
	c. Instructor Alerts Group	
	d. Environmental Conditions Group	
	e. Malfunction Status Group	
	Map Displays - Graph Area (pp. 65-73)	
1.	Air Speed Graph	Low
2.	Altitude Graph	Low
	Map Displays - Map Plot Area (pp. 65-73)	
1.	Cross Country Map	Average
2.	Approach Map	Average
3.	GCA Graph	Average
4.	GCA Information	Average
	a. Aircraft Identification	
	b. Heading	
	c. Position Relative to Course	
	d. Range	
	e. Altitude	
	Map Components (pp. 65-73)	
1.	Ground Track	Low

2.	Event Symbols	Low

Auxiliary Information Display (pp. 51-53)

1.	Flight Parameter List	Low
2.	Initial Condition Sets	Low
3.	Malfunction Tables	Low
4.	Radio Navigation Lists	Low
5.	Stored Plots	Low

Cockpit Indicator Display

Note: No outputs since this display repeats selected cockpit information for view by the instructor.

Cockpit Instrument Panel (pp. 82,83, SRS, Vol 1)

Note: The following warning indicators were counted as malfunctions:

- Engine Air Filter Light*
- RPM Warning Light*
- Fire Warning Indicator Light*

1.	Airspeed	Low
2.	Pitch	Low
3.	Bank	Low
4.	Pressure Altitude	Low
5.	Fuel Pressure	Low
6.	Fuel Quantity	Low
7.	Engine Oil Pressure	Low
8.	Engine Oil Temperature	Low
9.	Engine RPM	Low
10.	Rotor RPM	Low
11.	ID 998 Synchronizer Angle	Low
12.	Main Generator Load	Low
13.	DC Voltage	Average
	a. Battery	
	b. Main Generator	
	c. Standby Starter-Generator	
	d. Essential Bus	
	e. Nonessential Bus	
14.	Standby Generator Load	Low
15.	AC Voltage	Low
	a. AB	
	b. AC	
	c. BC	
16.	Master Caution Enable/Disable	Low
17.	Gas Producer RPM	Low
18.	Exhaust Gas Temperature	Low
19.	Vertical Velocity	Low
20.	Torque Pressure	Low
21.	Transmission Oil Pressure	Low
22.	Transmission Oil Temperature	Low

23.	Turn Rate	Low
24.	Slip	Low
25.	Magnetic Heading*	Low
26.	Outside Air Temperature*	Low

Note*: Not listed in Figure 3.4-25 of SRS but shown in Figure 3.4-23.

Trainer Status Information Display (p. 83, no's. 31-36, SRS, Vol 1)

Note: INSTR ACK was counted as an inquiry.

1.	PROB FRZ indicator light	Low
2.	MTN OFF indicator light	Low
3.	AUTO COPILOT indicator light	Low
4.	TRNR READY indicator light	Low
5.	PLAY BACK ON indicator light	Low

Problem Control Panel (pp. 90-92)

1.	SLOW indicator	Low
2.	IN PROG indicator	Low

Caution/Advisory Indicators (page 93 of SRS, Vol 1)

Note: The RESET/TEST illumination will be performed by stimulating the hardware directly.

Note: These were already counted as outputs (see Attachment A), or associated with a malfunction which was counted as an input.

- Engine Oil Pressure
- Engine Chip Detect
- Left Fuel Boost
- Right Fuel Boost
- Engine Fuel Pump
- 20 Minutes Fuel
- Fuel Filter
- Gov Emer
- Aux Fuel Low
- XMSN Oil Pressure
- XMSN Oil Hot
- Hydraulic Pressure
- Engine Inlet Air
- Inst Invertor
- DC Generator
- External Power
- Chip Detector
- IFF

Station Identifiers

1.	Marker Beacon Signal	Low

	Localizer Signal	Low
3.	VOR Signal	Low
4.	ADF Signal	Low

Other

Note: The temporary hydraulics malfunction provides feedback forces to the collective, cyclic, and pedals.

1.	Cyclic	Average
2.	Pedals	Average
3.	Collective	Average
4.	DC Circuit Breakers	Average
5.	AC Circuit Breakers	Low
6.	ADF Radio Tuning Meter Position	Low
7.	Touchdown/Crash Condition	Average

Total Number of UH-1 FS External Outputs

	Low	Average	High
TOTAL:	**47**	**10**	**1**

LOGICAL INTERNAL FILES

Key Points:

- A logical internal file is each logical group of data that is generated, used, and maintained by the application.
- Logical internal files are accessible to the user through external input, output or inquiry type.
- Databases are logical internal file types.
- The user must be aware that the file exists ie., the file is not implementation dependent.

Potential Types Within the UH-1 FS

- Runtime Data Bases

Description	**Complexity**
Runtime Data Bases	

Note: Malfunction Tables, Radio Navigation Lists, and Map Files were brought over via courseware files and cannot be edited. Therefore, they were not counted as Logical Internal Files.

	Description	Complexity
1.	Initial Condition Sets	Average
2.	Flight Parameters	Average
	Other	
1.	Stored Plots	Low
2.	Playback Information	High

Total Number of UH-1 FS External Interfaces

	Low	Average	High
Runtime Data Bases:	0	2	0
Other:	1	0	1
TOTAL:	**1**	**2**	**1**

EXTERNAL INQUIRIES

Key Points

- Each unique input output combination
- Cause and generate an immediate output
- Causes no change to internal data
- Do not count a soft key as an inquiry if it generates a picture that was counted as an external output.

Potential Types Within the UH-1 FS

- Graphic Display Controls
- Auxiliary Information Display Controls
- Test Switch
- Other

The following unique input/output combinations were counted for the UH-1 FS. The "input" part of the combination is numbered under the category heading, Input. The numbers correspond to values under the heading Result, to show each unique pair. The complexity of the input/output combination is listed with the input part of the combination.

	Description	Complexity
INPUT		
	Timer/Display Control Panel (pp. 62-65)	
1.	Cockpit Display Select Controls	Low
	a. 1 push button	
	b. 2 push button	
2.	Display to Student	Low
	a. 1 push button	
	b. 2 push button	
3.	Timer	Low
	a. Start/Stop	
	b. Reset	
	Trainee Station Control Panel (pp. 38-40)	
	Graphic Display Controls:	
1.	Scaling push buttons	Low
	a. G TRK 12.5 x 12.5	
	b. G TRK 25 x 25	
	c. G TRK 100 x 100	
2.	FULL SCALE AS	Low

3. FULL SCALE ALT — Low

4. Hard Copy (p. 34)
   a. PRINT PLTR SMY — Low
   b. PRINT PROC SMY — Low

<u>Auxiliary Information Display (AID) Control Panel</u> (pp 43-53)

Note: The following Display Select Controls -
   a. GCA push button
   b. CROSS CNTRY push button
   c. AREA push button, and
   Display/Edit Format Select Controls -
   a. FLT PARAM push button
   b. FAIL push button
   c. INIT COND push button
   d. STORED PLOTS push button
   e. RADIO NAV push button
   were counted as outputs.

1. Display Area Select Controls — Low
   a. EDIT AREA push button
   b. CKPT 1 AREA push button
   c. CKPT 2 AREA push button
   d. CKPT 3 AREA push button
   e. CKPT 4 AREA push button

2. Transfer Cockpit Area to Edit Area Controls — Low
   a. 1 push button
   b. 2 push button
   c. 3 push button
   d. 4 push button

<u>Test Switch</u>

1. Fire Detector Test Switch (p. 82) — Low
2. Fuel Gauge Test Switch (pp. 82, 86) — Low
3. Chip Detector Switch (P. 89) — Low

<u>Problem Control Panel</u>

1. INSTR CALL push button — Low

<u>IFF Transponder Set Control Panel</u> (MMR, p. 181)

1. Transponder Master Control Switch — Low

<u>AC Power Panel</u> (p. 94)

1. AC Power Phase selector — Low

DC Power Panel (p. 99)

1. DC Voltmeter selector switch — Low

RESULT

Timer/Display Control Panel

1. Repeater Instruments show readings for the selected cockpit. Cockpit Select Indicators will be illuminated based on which selection was made at the Timer/Display Control Panel.
2. Allows Cockpit CRT display within a cockpit.
3. Timer display is controlled by Start/Stop and Reset buttons.

Trainee Station Control Panel

1. Approach Map is displayed at the selected scale.
2. Airspeed Graph is enlarged.
3. Altitude Graph is enlarged.
4. (unknown)

Auxiliary Information Display Controls

1. Allows selection of cockpit areas in the display area of the AID.
2. Transfers a display from the display area of the AID to the edit area of the AID.

Test Switch

1. Causes Fire Warning Light to Illuminate while depressed.
2. Causes Fuel Quantity Indicator to move from actual reading to lesser reading.
3. Indicates the trouble area when the Chip Detector caution light is illuminated.

Problem Control Panel

1. ACK STUD push button flashes.

IFF Transponder Set Control Panel

1. Transponder mode is displayed at the Problem Status Display.

AC Power Panel

1. Permits monitoring of any one of the three phases (AB, AC, and BC) on the AC Voltmeter.

DC Power Panel

1. Permits monitoring of voltage being delivered from any of the following sources: Battery, Main

Generator, Standby Starter-Generator, Essential Bus, Non-Essential Bus.

Total Number of UH-1 FS External Inquiries

	Low	Average	High
TOTAL:	**17**	**0**	**0**

EXTERNAL INTERFACES

Key Points:

- Files passed or shared between applications should be counted as external interface types

Potential Types Within the UH-1 FS

- Data files from the ACS

	Description	**Complexity**
1.	Courseware Files	High

Total Number of UH-1 FS External Interfaces

	Low	Average	High
Other:	0	0	1
TOTAL:	**0**	**0**	**1**

FUNCTION POINTS CALCULATION

Function Count:

Description

	Low	Average	High	Total
External Input	60 x 3 = 180	11 x 4 = 44	1 x 6 = 6	230
External Output	47 x 4 = 188	10 x 5 = 50	1 x 7 = 7	245
Logical Internal	1 x 7 = 7	2 x 10 = 20	1 x 15 = 15	42
Ext Interface File	0 x 5 = 0	0 x 7 = 0	1 x 10 = 10	10
External Inquiry	17 x 3 = 51	0 x 4 = 0	0 x 16 = 0	51

Total Unadjusted Function Points (FC): 578

General Information Processing Function

	Characteristic	Degree of Influence Value
1.	Data Communications	4
2.	Distributed Functions	5
3.	Performance	4
4.	Heavily Used Configuration	3
5.	Transaction Rate	3
6.	Online Data Entry	1
7.	End User Efficiency	3
8.	Online Update	4
9.	Complex Processing	3
10.	Reusability	2
11.	Installation Ease	4
12.	Operational Ease	3
13.	Multiple Sites	4
14.	Facilitate Change	1
	Total Degree of Influence (TDI):	44

GCA	**General Info. Proc. Func. Adj.**	= 0.65 + (0.01 x TDI)	= 1.09
FP	**Function Points Measure**	= FC x GCA	= 630.02
LEX	**Ada Language Expansion Factor**	=	= 71
SLOC	**Source Lines of Code Count**	= FP x LEX	= 44,731

ATTACHMENT A

EXTERNAL OUTPUTS FOR COCKPIT INSTRUMENT PANEL

	Output	Output Device
1.	AC Voltage	• AC Voltmeter • Inst Invertor Caution Light
2.	Pitch	• Attitude Indicator
3.	Bank	• Attitude Indicator
4.	Airspeed	• Airspeed Indictor • Airspeed Linear Indicator
5.	DC Voltage	• DC Voltmeter
6.	Rotor RPM	• Dual Tachometer • Rotor RPM Linear Indicator
7.	Engine RPM	• Dual Tachometer • Engine RPM Linear Indicator • Low RPM Audio
8.	Engine Oil Pressure	• Engine Oil Pressure Indicator • Engine Oil Pressure Caution Light
9.	Engine Oil Temperature	• Engine Oil Temperature Indicator
10.	Exhaust Gas Temperature	• Exhaust Gas Temperature Indicator • Exhaust Gas Temperature Linear Indicator
11.	Fuel Quantity	• Fuel Quantity Indicator • Minutes of Fuel Remaining Digital Readout • 20 Minute Fuel Remaining Caution Light • Auxiliary Fuel Low Caution Light
12.	Fuel Pressure	• Fuel Pressure Indicator
13.	Gas Producer RPM	• Gas Producer Tachometer • Gas Producer Linear Indicator
14.	Main Generator Load	• Main Generator Loadmeter
15.	Magnetic Heading	• Radio Magnetic Indicator
16.	Master Caution Enable/Disable	• Master Caution Light
17.	Outside Air Temperature	• Outside Air Temperature Indicator
18.	Pressure Altitude	• Barometric Pressure Altimeter • Altitude Digital Readout
19.	ID 998 Synchronizer Angle	• Radio Magnetic Indicator
20.	Standby Generator Load	• Standby Generator Loadmeter
21.	Torque Pressure	• Torquemeter • Torque Pressure Linear Indicator
22.	Transmission Oil Pressure	• Transmission Oil Pressure Indicator • Transmission Oil Pressure Caution Light
23.	Transmission Oil Temperature	• Transmission Oil Temperature Indicator • Transmission Oil Hot Caution Light
24.	Slip	• Turn & Slip Indicator
25.	Turn Rate (Yaw)	• Turn & Slip Indicator
26.	Vertical Velocity	• Vertical Velocity Indicator • Vertical Velocity Linear Indicator

APPENDIX E

DERIVATION OF FUNCTION POINT COUNT FOR THE AUTOMATED COURSEWARE SYSTEM

This attachment contains a description of how function point parameters were counted for the Automated Courseware System (ACS). The Automated Courseware System (ACS) software is the component providing the capability to develop and modify trainer courseware via the Automated Courseware workstation. The ACS provides for the formulation and editing of UH-1 FS mission scenarios consisting of navigational aides, initial operating conditions, and real-time maps.

The ACS is divided into two CSCs: 1) the ACS CSC, and 2) Map Preview CSC. The ACS CSC is the software for all the courseware data entry tasks. It is activated from the system main menu whenever courseware modifications are to be made. Running the ACS CSC results in the creation of a courseware floppy. The Courseware Loader must be run to read the ACS courseware floppy and create real-time format data tables. Having loaded the courseware files, the Map Preview CSC can be used to view the map displays. When viewing is complete, pressing Control-Q (^ Q) quits the program and returns the user to the ACS main menu.

Counting conventions are presented by parameter type:

- External Inputs
- External Outputs
- Logical Internal Files
- External Inquiries
- External Interfaces.

For each parameter type the description of how counts were derived contains the following information:

- **Key points** - a summary of the basic parameter definition with emphasis of certain key factors.

- **Potential types within the UH-1 FS** - situations in which elements were counted as this parameter type.

- **Description** - an annotated listing of each parameter that was counted and the complexity level that was assigned.

- **Total number of element types** - total count of elements for the specified parameter.

EXTERNAL INPUTS

Key Points:

- User data or user control information that enters the external boundary of the application
- It must change something inside the system
- It is unique if it has a different format or requires different processing logic

Navaids Data State Menus

	Description	**Complexity**
1.	Navaids Data Menu	Low
2.	ADF Menu	Low
3.	VOR Menu	Low
4.	TAC Menu	Low
5.	VORTAC Menu	Low
6.	ILS Menu	Average
7.	GCA Menu	Average
8.	UHF Menu	Low
9.	FM Menu	Low
10.	VHF Menu	Low
11.	Intersections Menu	Low
12.	Obstructions Menu	Low
13.	Vector Menu	Average

Training Data State Menus

	Description	**Complexity**
1.	Training Data Menu	Low
2.	Initial Conditions Data Edit Menu	Average
3.	Approach Map Specifications Edit Menu	Low
4.	Gaming Area Specifications Edit Menu	Low
4.	Navaids Selection Menu	Low

EXTERNAL OUTPUTS

Key Points:

- User data or user control information that leaves the external boundary of the application measured
- It is unique if it has a different format or requires different processing logic
- It does not include output response of an external inquiry

	Description	Complexity
	Reports	
1.	Navaids data base sorted by airfield and call sign designation	Low
2.	Training area data (IC, map, and selected navaids data)	Average
	Map Preview Graphic Displays	
1.	Cross County Map (Map #0)	Average
2.	Approach Plate Maps (Map #0 - #9)	Average

LOGICAL INTERNAL FILES

Key Points:

- A logical internal file is each logical group of data that is generated, used, and maintained by the application.
- Logical internal files are accessible to the user through external input, output or inquiry type
- Databases are logical internal file types
- The user must be aware that the file exists ie., the file is not implementation dependent

ACS Data Base

	Description	Complexity
1.	Navaids	Average
2.	Initial Condition Sets	Low
3.	Map Definitions	Low
4.	Gaming Area Definitions	Low

EXTERNAL INQUIRIES

Key Points:

- Each unique input - output combination
- Cause and generate an immediate output
- Causes no change to internal data
- Do not count a soft key as an inquiry if it generates a picture that was counted as an external output

Control Keys

	Description	Complexity
1.	Finished (^F)	Low
2.	Page (^P)	Low
3.	Quit (^Q)	Low

Menus

1.	ACS Main Menu	Low

EXTERNAL INTERFACES

Key Points:

- Files passed or shared between applications should be counted as external interface types

Hardware/Software Interfaces

	Description	Complexity
1.	Courseware Files	Average

FUNCTION POINTS CALCULATION

Function Count:

Description

	Low	Average	High	Total
External Input	14 x 3 = 42	4 x 4 = 16	0 x 6 = 0	58
External Output	1 x 4 = 4	3 x 5 = 15	0 x 7 = 0	19
Logical Internal	3 x 7 = 21	1 x 10 = 10	0 x 15 = 0	31
Ext Interface	0 x 5 = 0	1 x 7 = 7	0 x 10 = 0	7
Ext Inquiry	4 x 3 = 12	0 x 4 = 0	0 x 6 = 0	12

Total Unadjusted Function Points (FC): 127

General Information Processing Function

	Characteristic	Degree of Influence Value
1.	Data Communications	0
2.	Distributed Functions	2
3.	Performance	0
4.	Heavily Used Configuration	0
5.	Transaction Rate	0
6.	Online Data Entry	5
7.	End User Efficiency	2
8.	Online Update	3
9.	Complex Processing	0
10.	Reusability	1
11.	Installation Ease	2
12.	Operational Ease	1
13.	Multiple Sites	0
14.	Facilitate Change	0
	Total Degree of Influence (TDI):	**16**

CAF	***Complexity Adjustment Factor***	= 0.65 + (0.01 x TDI)	= .81
FP	**Function Points Measure**	= FC x CAF	= 102.87
LEX	***FORTRAN* Language Expansion Factor**	=	= 71
SLOC	**Source Lines of Code Count**	= FP x LEX	= 7,304

(This Page is Intentionally Left Blank)

APPENDIX F

RESOURCE ESTIMATION REPORTS

The NASA Interactive Software Cost Modeling System
Version 5.1.0
January 15, 1992 16:46:45

PROJECT TITLE
ADA COCOMO, from Carnegie Mellon conference

PROJECT DATA FILE

File Title: UH-1 FS Redevelopment Project - Estimate at completion
File Name: UH1FS Save Date: 1/15/1992 Project Name: UH1FS

PROJECT CONFIGURATION FILES

Phase Distribution Data File:	P871104A
Effort & Schedule Data File	EFFSCH89
Multiplier Data File	ADACOCMO

Estimating Model: ADA COCOMO Project Mode: Semi-Detached

PROJECT DEVELOPMENT COSTING EQUATION

Effort Coefficient : 3.00	Effort Exponent : 1.10
Schedule Coefficient: 3.00	Schedule Exponent: 0.38

PROJECT MAINTENANCE COSTING EQUATION

Effort Coefficient : 3.00	Effort Exponent : 1.05

Data For The ADA COCOMO Estimation Equation
ADA COCOMO, from Carnegie Mellon conference
January 15, 1992 16:46:45

Development	Ratings	Maintenance	Ratings
Exp W/ ADA	0.040	Use Of MPPs	0.030
Design At PDR	0.034	Conformance	0.020
Risks By PDR	0.040		
Req Volatility	0.016		
(Sigma)Summation Of Rates	0.130	Summation Of Rates	0.050

Difference between EMBEDDED MODE and ADA MODE with ratings of Zero = 0.160

User Selections that make up the ratings for the Estimating Equations
The Current Selections for this case are shown in bold type

The values related to the respective selection columns are:
0.0 0.01 0.02 0.03 0.04 0.05

Selections which comprise the "Exp W/ ADA" rating

	0.0	0.01	0.02	0.03	0.04	0.05
Exp W/ Ada	Greatest	Greater	General	Some	**Little**	No

Selections which comprise the "Design at PDR" rating

	0.0	0.01	0.02	0.03	0.04	0.05
Sch,Budget,Etc	Fully	Mostly	Generally	Some	Little	**None**
% of Dev Sch	40%	33%	25%	17%	**10%**	5%
% Req Top S/W	120%	100%	80%	**60%**	40%	20%
Tool Support	Full	Strong	Good	**Some**	Little	None
Level Uncert	Very Little	Little	**Some**	Consid- erable	Signif- icant	Extreme

Selections which comprise the "Risks by PDR" rating

Risk Management	Fully	Mostly	Generally	Some	Little	**None**
Sch,Budget,Etc	Fully	Mostly	Generally	Some	Little	**None**
% of Dev Sch	40%	33%	25%	17%	**10%**	5%
% Req Top S/W	120%	100%	80%	**60%**	40%	20%
Tool Support	Full	Strong	Good	**Some**	Little	None

Selections which comprise the "Req Volatility" rating

Sys Req Base	**Fully**	Mostly	Generally	Some	Little	None
Level Uncert	Very Little	**Little**	Some	Consid-erable	Signif-icant	Extreme
Org Track Recd	Excellent	**Strong**	Good	Some	Little	None
Use of Incr Dev	Full	Strong	Good	**Some**	Little	None
Sys Arch Mod	Fully	Mostly	Generally	**Some**	Little	None

Selections which comprise the "Use of MPPs" rating

Use of MPPs	Greatest	**Greater**	General	Some	Little	No

Selections which comprise the "Conformance" rating

Maint Conform	Full	General	**Often**	Some	Little	None

ADA COCOMO PROJECT DEFINITION DATA
ADA COCOMO, from Carnegie Mellon conference
January 15, 1992 16:46:45

Component Number 1 Component Name: UH1 FS

LINES OF NEW CODE		ADAPTED CODE	
Least :	78K	Total Lines :	13K
Most Likely:	78K	Percentage of ReDesign :	10%
Greatest :	78K	Percentage of ReCode :	30%
		Percentage of Integration :	10%

Dollar Cost per Man Month: $ 0

ONGOING MAINTENANCE
Lines Added per Year: 0K Lines Modified per Year: 0K

COST DRIVER RATINGS AND VALUES

COST DRIVER	RELY	DATA	CPLX	RUSE	TIME
Development	LO 0.88	LO 0.94	HI 1.08	VH 1.30	NO 1.00
Maintenance	NO 0.96	NO 1.00	NO 0.97	NO 1.00	NO 1.00

COST DRIVER	STOR	VMVH	VMVT	TURN	ACAP
Development	NO 1.00	LO 0.92	LO 0.93	LO 0.87	HI 0.80
Maintenance	NO 1.00	NO 1.00	NO 1.00	NO 1.00	NO 1.00

COST DRIVER	PCAP	AEXP	VEXP	LEXP	MODP
Development	NO 1.00	NO 1.00	NO 1.00	LO 1.14	NO 0.98
Maintenance	NO 1.00	NO 1.00	NO 1.00	NO 1.04	NO 1.00

COST DRIVER	TOOL	SECU
Development	NO 1.00	NO 1.00
Maintenance	NO 1.00	NO 1.00

Schedule Cost Driver Eaf = 1.00
Remaining Cost Driver Eaf = 0.77
Total Cost Driver Eaf = 0.77

Software Development Costs Using
Semi-Detached Mode

ADA COCOMO, from Carnegie Mellon conference
January 15, 1992 16:46:45

Component	KEDSI	AAF	EAF	MM Nom	MM Dev	EDSI/MM	K$/Comp	$/EDSI
1- UH1 FS	79.9	16	0.77	374	289	277	0	0
Total KEDSI	79.9		Totals		289	277	0	0
(MM) Nom	374	Dev Schedule			25.2 Nom schedule			25.2
(EDSI/MM)-Nom	214							

ADA COCOMO PROJECT DEFINITION DATA
ADA COCOMO, from Carnegie Mellon conference
January 15, 1992 16:46:45

LINES OF NEW CODE		ADAPTED CODE	
Least	: 77.765K	Total Lines	: 13.472K
Most Likely	: 77.765K	Percentage of ReDesign	: 10%
Greatest	: 77.765K	Percentage of ReCode	: 30%
		Percentage of Integration	: 10%

Dollar Cost per Man Month: $0

ONGOING MAINTENANCE
Lines Added per Year: 0K Lines Modified per Year: 0K

O U T P U T S U M M A R Y
ADA COCOMO, from Carnegie Mellon conference
January 15, 1992 16:46:45

ACTIVITY DISTRIBUTION BY PHASE

Total Delivered Source Instructions:	79.9K
Effective Delivered Source Instructions per Man Month:	276.9
Number of FullTime Software Professionals:	11

Product Development Effort = 288.7 Man Months

ACTIVITY	Plans & Reqmnts M Months	Plans & Reqmnts (%)	Product Design M Months	Product Design (%)	Programming M Months	Programming (%)	Integrat & Test M Months	Integrat & Test (%)	Maintenance M Months	Maintenance (%)
Requirements Analyses	9.2	45.5	6.1	12.5	6.5	4.0	1.9	2.5	0.0	0.0
Product Design	3.5	17.2	20.1	41.0	13.0	8.0	3.8	5.0	0.0	0.0
Programming	1.0	5.0	6.5	13.2	92.2	56.5	29.1	38.0	0.0	0.0
Test Planning	0.8	3.7	2.8	5.7	8.6	5.2	2.3	3.0	0.0	0.0
Verification & Validation	1.5	7.2	3.6	7.2	13.5	8.2	22.2	29.0	0.0	0.0
Project Office	2.6	13.0	5.2	10.5	10.2	6.3	5.5	7.3	0.0	0.0
Conf Mgt/Quality Assurance	0.6	3.0	1.2	2.5	10.6	6.5	6.1	8.0	0.0	0.0
Manuals	1.1	5.3	3.6	7.3	8.6	5.3	5.5	7.3	0.0	0.0
TOTALS	20.2	7.0	49.1	17.0	163.1	56.5	76.5	26.5	0.0	0.0

Product Development Costs = 0.0 K Dollars

ACTIVITY	Plans & Reqmnts K Dollars	Plans & Reqmnts FSP	Product Design K Dollars	Product Design FSP	Programming K Dollars	Programming FSP	Integrat & Test K Dollars	Integrat & Test FSP	Maintenance K Dollars	Maintenance FSP
Requirements Analyses	0	1.7	0	0.9	0	0.6	0	0.3	0	0.0
Product Design	0	0.7	0	3.0	0	1.1	0	0.6	0	0.0
Programming	0	0.2	0	1.0	0	7.9	0	4.2	0	0.0
Test Planning	0	0.1	0	0.4	0	0.7	0	0.3	0	0.0
Verification & Validation	0	0.3	0	0.5	0	1.2	0	3.2	0	0.0
Project Office	0	0.5	0	0.8	0	0.9	0	0.8	0	0.0
Conf Mgt/Quality Assurance	0	0.1	0	0.2	0	0.9	0	0.9	0	0.0
Manuals	0	0.2	0	0.5	0	0.7	0	0.8	0	0.0
TOTALS	0	3.8	0	7.3	0	14.0	0	11.0	0	0.0

Product Development Schedule = 25.2 Months

ACTIVITY	Plans & Reqmnts Months	Plans & Reqmnts (%)	Product Design Months	Product Design (%)	Programming Months	Programming (%)	Integrat & Test Months	Integrat & Test (%)	Maintenance Months	Maintenance (%)
Requirements Analyses	2.4	45.5	0.8	12.5	0.5	4.0	0.2	2.5	0.0	0.0
Product Design	0.9	17.2	2.7	41.0	0.9	8.0	0.3	5.0	0.0	0.0
Programming	0.3	5.0	0.9	13.2	6.6	56.5	2.6	38.0	0.0	0.0
Test Planning	0.2	3.7	0.4	5.7	0.6	5.2	0.2	3.0	0.0	0.0
Verification & Validation	0.4	7.2	0.5	7.2	1.0	8.2	2.0	29.0	0.0	0.0
Project Office	0.7	13.0	0.7	10.5	0.7	6.3	0.5	7.3	0.0	0.0
Conf Mgt/Quality Assurance	0.2	3.0	0.2	2.5	0.8	6.5	0.6	8.0	0.0	0.0
Manuals	0.3	5.3	0.5	7.3	0.6	5.3	0.5	7.3	0.0	0.0
TOTALS	5.3	21.0	6.7	26.5	11.6	46.0	6.9	27.5	0.0	0.0

PROJECT INFORMATION

Project name	:	PMTRADE
Estimate date and time	:	01/16/92 09:16 am
Version	:	pmtrade - final
Start date	:	01/30/92
Number of Subprojects	:	0

CALIBRATION COEFFICIENTS

Productivity multiplier (A)	:	1.420
Schedule multiplier (B)	:	3.000
Effort exponent (alpha)	:	1.200
Schedule exponent (beta)	:	0.400
Base effort constant (gamma)	:	2.600
Work hours/person-month	:	160.0

SIZING SUBMODEL WEIGHTING FACTORS

New Ada Components	:	1.000
Reused Ada Components	:	0.200
Modified Ada Components	:	0.300
New Other Components	:	1.000
Reused Other Components	:	0.250
Modified Other Components	:	0.400

	Effective Size (KSLOC/FP)	Duration (months)	Effort (pm)	Productivity (SLOC/pm)	Average Staff (persons)	Confidence (%)
PMTRADE	50.6	33.3	410.0	132.0	12.3	54.6

PMTRADE	PROJECT FACTORS	
Type of Software	Simulation	
System Architecture	Loosely-coupled	(1.300)
Number of Organizations Involved	5	
Organizational Interface Complexity	High	
Staff Resource Availability	Nominal	(1.000)
Computer Resource Availability	Nominal	(1.000)
Security Requirements	Low	(0.950)

PMTRADE	PROCESS FACTORS	
Degree of Standardization	Very High	(1.290)
Scope of Support	Low	(0.950)
Use of Modern Software Methods	Nominal	
Use of Peer Reviews	Very High	(1.650)
Use of Software Tools/Environment	Low	
Software Tool/Environment Stability	Very High	

PMTRADE	PRODUCT FACTORS	
Ada Usage Factor	High	(1.080)
Product Complexity	High	(1.160)
Requirements Volatility	Nominal	(1.000)
Degree of Optimization	Low	(0.910)
Degree of Real-Time	High	(1.110)
Reuse Benefits	Nominal	
Reuse Costs	Low	
Database Size	Low	(0.940)

PMTRADE	PERSONNEL FACTORS	
Number of Ada Projects Completed	0	
Analyst Capability	High	(0.890)
Applications Experience	Nominal	(1.000)
Ada Environment Experience	Very Low	(1.250)
Ada Language Experience	Low	(1.190)
Ada Methodology Experience	High	(0.890)
Team Capability	Very High	(0.850)

KILO-LINES OF SOURCE CODE

PMTRADE	MAX	MOST LIKELY	MIN	WEIGHTED
New Ada Components	42.1	42.1	42.1	42.1
Reused Ada Components	6.2	6.2	6.2	0.7
Modified Ada Components	3.0	3.0	3.0	0.9
New Other Components	7.0	7.0	7.0	7.0
Reused Other Components	0.1	0.1	0.1	0.0
Modified Other Components	0.1	0.1	0.1	0.0

Total effective size : 50.7 KSLOC Size variance: 0.0 KSLOC

ADA OBJECT-ORIENTED PARADIGM

Life Cycle Phases	SW Development Effort	SW Management Effort	SCM Effort	SQE Effort
System Reqts Analysis/Design	6232.0	623.2	311.6	311.6
SW Requirements Analysis	12464.0	1246.4	467.4	623.2
SW Preliminary Design	9348.0	934.8	467.4	934.8
SW Detailed Design	9348.0	934.8	623.2	934.8
Coding and CSU Testing	9348.0	934.8	623.2	934.8
CSC Integration & Testing	15580.0	1558.0	934.8	1246.4
CSCI Testing	6232.0	623.2	623.2	623.2
System Integration & Testing	18696.0	1869.6	311.6	623.2
	--------	--------	--------	--------
TOTALS	87248.0	8724.8	4362.4	6232.0

PROJECT: PMTRADE EFFORT: 410.0 PERSON-MONTHS DURATION: 33.3 MONTHS

CODE	TASK	DUR	EFFORT	START EARLY DAY	START EARLY DATE	START LATE DAY	START LATE DATE	FINISH EARLY DAY	FINISH EARLY DATE	FINISH LATE DAY	FINISH LATE DATE	SLACK TIME DAYS
1.	SYS REQT ANALYS/DSGN PHASE	0.0	0.0	0	01DEC87	0	01DEC87	0	01DEC87	0	01DEC87	0
1.1	SW DEVELOPMENT TASKS 1	0.0	0.0	0	01DEC87	0	01DEC87	0	01DEC87	0	01DEC87	0
1.1.1	SUPPORT SS DEV	26.6	41.0	0	01DEC87	0	01DEC87	27	11JAN88	27	11JAN88	0
1.1.2	SUPPORT OCD DEV	26.6	41.0	0	01DEC87	0	01DEC87	27	11JAN88	27	11JAN88	0
1.1.3	SUPPORT SRR	6.7	41.0	27	11JAN88	27	11JAN88	33	20JAN88	33	20JAN88	0
1.1.4	SUPPORT SSDD DEV	20.0	164.0	33	20JAN88	33	20JAN88	53	19FEB88	53	19FEB88	0
1.1.5	DEVELOP PREL SRS(S)	20.0	287.0	33	20JAN88	33	20JAN88	53	19FEB88	53	19FEB88	0
1.1.6	DEVELOP PREL IRS(S)	20.0	164.0	33	20JAN88	33	20JAN88	53	19FEB88	53	19FEB88	0
1.1.7	SUPPORT STP DEV	20.0	41.0	33	20JAN88	33	20JAN88	53	19FEB88	53	19FEB88	0
1.1.8	SUPPORT SDR	13.3	41.0	53	19FEB88	53	19FEB88	67	11MAR88	67	11MAR88	0
1.1.9	SDR COMPLETED	0.0	0.0	67	11MAR88	67	11MAR88	67	11MAR88	67	11MAR88	0
1.2	SW MANAGEMENT TASKS 1	0.0	0.0	0	01DEC87	0	01DEC87	0	01DEC87	0	01DEC87	0
1.2.1	PREPARE SDP	66.6	82.0	0	01DEC87	0	01DEC87	67	11MAR88	67	11MAR88	0
1.3	SCM TASKS 1	0.0	0.0	0	01DEC87	0	01DEC87	0	01DEC87	0	01DEC87	0
1.3.1	PREPARE CM PLAN	66.6	41.0	0	01DEC87	0	01DEC87	67	11MAR88	67	11MAR88	0
1.4	SQE TASKS 1	0.0	0.0	0	01DEC87	0	01DEC87	0	01DEC87	0	01DEC87	0
1.4.1	PREPARE QE PLAN	66.6	41.0	0	01DEC87	0	01DEC87	67	11MAR88	67	11MAR88	0
2.	SW REQTS ANALYSIS PHASE	0.0	0.0	67	11MAR88	67	11MAR88	67	11MAR88	67	11MAR88	0
2.1	SW DEVELOPMENT TASKS 2	0.0	0.0	67	11MAR88	67	11MAR88	67	11MAR88	67	11MAR88	0
2.1.1	DEVELOP SRS(S)	113.2	410.0	67	11MAR88	67	11MAR88	180	26AUG88	180	26AUG88	0
2.1.2	DEVELOP IRS(S)	113.2	246.0	67	11MAR88	67	11MAR88	180	26AUG88	180	26AUG88	0
2.1.3	DEVELOP PREL MANUALS	79.9	164.0	67	11MAR88	100	28APR88	147	11JUL88	180	26AUG88	33
2.1.4	ACQUIRE/READY SEE	113.2	410.0	67	11MAR88	67	11MAR88	180	26AUG88	180	26AUG88	0
2.1.5	CONDUCT TRAINING	66.6	164.0	67	11MAR88	113	18MAY88	133	17JUN88	180	26AUG88	47
2.1.6	CONDUCT SSR	20.0	246.0	180	26AUG88	180	26AUG88	200	27SEP88	200	27SEP88	0
2.1.7	SSR COMPLETED	0.0	0.0	200	27SEP88	200	27SEP88	200	27SEP88	200	27SEP88	0
2.2	SW MANAGEMENT TASKS 2	0.0	0.0	67	11MAR88	87	11APR88	67	11MAR88	87	11APR88	20
2.2.1	CONDUCT REVIEWS 2	113.2	164.0	67	11MAR88	87	11APR88	180	26AUG88	200	27SEP88	20
2.3	SCM TASKS 2	0.0	0.0	67	11MAR88	67	11MAR88	67	11MAR88	67	11MAR88	0
2.3.1	DEVELOP CM PROC	33.3	20.5	67	11MAR88	67	11MAR88	100	28APR88	100	28APR88	0
2.3.2	OPERATE SW LIBRARY 2	99.9	41.0	100	28APR88	100	28APR88	200	27SEP88	200	27SEP88	0
2.4	SQE TASKS 2	0.0	0.0	67	11MAR88	67	11MAR88	67	11MAR88	67	11MAR88	0
2.4.1	DEVELOP QE PROC	33.3	41.0	67	11MAR88	67	11MAR88	100	28APR88	100	28APR88	0
2.4.2	CONDUCT EVALUATIONS 2	99.9	41.0	100	28APR88	100	28APR88	200	27SEP88	200	27SEP88	0
3.	SW PREL DESIGN PHASE	0.0	0.0	200	27SEP88	200	27SEP88	200	27SEP88	200	27SEP88	0
3.1	SW DEVELOPMENT TASKS 3	0.0	0.0	200	27SEP88	200	27SEP88	200	27SEP88	200	27SEP88	0
3.1.1	DEVELOP SDD(S)	79.9	410.0	200	27SEP88	200	27SEP88	280	27JAN89	280	27JAN89	0
3.1.2	DEVELOP IDD(S)	79.9	246.0	200	27SEP88	200	27SEP88	280	27JAN89	280	27JAN89	0
3.1.3	DEVELOP STP	79.9	328.0	200	27SEP88	200	27SEP88	280	27JAN89	280	27JAN89	0

PROJECT: PMTRADE EFFORT: 410.0 PERSON-MONTHS DURATION: 33.3 MONTHS

CODE	TASK	DUR	EFFORT	START EARLY DAY	START EARLY DATE	START LATE DAY	START LATE DATE	FINISH EARLY DAY	FINISH EARLY DATE	FINISH LATE DAY	FINISH LATE DATE	SLACK TIME DAYS
3.1.4	CONDUCT PDR	20.0	246.0	280	27JAN89	280	27JAN89	300	01MAR89	300	01MAR89	0
3.1.5	PDR COMPLETED	0.0	0.0	300	01MAR89	300	01MAR89	300	01MAR89	300	01MAR89	0
3.2	SW MANAGEMENT TASKS 3	0.0	0.0	200	27SEP88	226	07NOV88	200	27SEP88	226	07NOV88	27
3.2.1	CONDUCT REVIEWS 3	73.3	123.0	200	27SEP88	226	07NOV88	273	18JAN89	300	01MAR89	27
3.3	SCM TASKS 3	0.0	0.0	200	27SEP88	200	27SEP88	200	27SEP88	200	27SEP88	0
3.3.1	OPERATE SW LIBRARY 3	99.9	61.5	200	27SEP88	200	27SEP88	300	01MAR89	300	01MAR89	0
3.4	SQE TASKS 3	0.0	0.0	200	27SEP88	200	27SEP88	200	27SEP88	200	27SEP88	0
3.4.1	CONDUCT EVALUATIONS 3	99.9	123.0	200	27SEP88	200	27SEP88	300	01MAR89	300	01MAR89	0
4.	SW DETAILED DESIGN PHASE	0.0	0.0	300	01MAR89	300	01MAR89	300	01MAR89	300	01MAR89	0
4.1	SW DEVELOPMENT TASKS 4	0.0	0.0	300	01MAR89	300	01MAR89	300	01MAR89	300	01MAR89	0
4.1.1	DETAIL SDD(S)	79.9	410.0	300	01MAR89	300	01MAR89	380	27JUN89	380	27JUN89	0
4.1.2	DETAIL IDD(S)	79.9	164.0	300	01MAR89	300	01MAR89	380	27JUN89	380	27JUN89	0
4.1.3	DEVELOP TEST CASE(S)	79.9	246.0	300	01MAR89	300	01MAR89	380	27JUN89	380	27JUN89	0
4.1.4	CONDUCT PEER REVIEWS 4	66.6	164.0	300	01MAR89	313	20MAR89	366	07JUN89	380	27JUN89	13
4.1.5	CONDUCT CDR	20.0	246.0	380	27JUN89	380	27JUN89	400	27JUL89	400	27JUL89	0
4.1.6	CDR COMPLETED	0.0	0.0	400	27JUL89	400	27JUL89	400	27JUL89	400	27JUL89	0
4.2	SW MANAGEMENT TASKS 4	0.0	0.0	300	01MAR89	326	07APR89	300	01MAR89	326	07APR89	27
4.2.1	CONDUCT REVIEWS 4	73.3	123.0	300	01MAR89	326	07APR89	373	16JUN89	400	27JUL89	27
4.3	SCM TASKS 4	0.0	0.0	300	01MAR89	300	01MAR89	300	01MAR89	300	01MAR89	0
4.3.1	OPERATE SW LIBRARY 4	99.9	82.0	300	01MAR89	300	01MAR89	400	27JUL89	400	27JUL89	0
4.4	SQE TASKS 4	0.0	0.0	300	01MAR89	300	01MAR89	300	01MAR89	300	01MAR89	0
4.4.1	CONDUCT EVALUATIONS 4	99.9	123.0	300	01MAR89	300	01MAR89	400	27JUL89	400	27JUL89	0
5.	CODING & CSU TESTING PHASE	0.0	0.0	400	27JUL89	400	27JUL89	400	27JUL89	400	27JUL89	0
5.1	SW DEVELOPMENT TASKS 5	0.0	0.0	400	27JUL89	400	27JUL89	400	27JUL89	400	27JUL89	0
5.1.1	CODE & TEST UNITS	79.9	820.0	400	27JUL89	400	27JUL89	480	28NOV89	480	28NOV89	0
5.1.2	CONDUCT PEER REVIEWS 5	79.9	246.0	400	27JUL89	400	27JUL89	480	28NOV89	480	28NOV89	0
5.1.3	CONDUCT UTR	20.0	164.0	480	28NOV89	480	28NOV89	500	27DEC89	500	27DEC89	0
5.1.4	UTR COMPLETED	0.0	0.0	500	27DEC89	500	27DEC89	500	27DEC89	500	27DEC89	0
5.2	SW MANAGEMENT TASKS 5	0.0	0.0	400	27JUL89	426	06SEP89	400	27JUL89	426	06SEP89	27
5.2.1	CONDUCT REVIEWS 5	73.3	123.0	400	27JUL89	426	06SEP89	473	15NOV89	500	27DEC89	27
5.3	SCM TASKS 5	0.0	0.0	400	27JUL89	400	27JUL89	400	27JUL89	400	27JUL89	0
5.3.1	OPERATE SW LIBRARY 5	99.9	82.0	400	27JUL89	400	27JUL89	500	27DEC89	500	27DEC89	0
5.4	SQE TASKS 5	0.0	0.0	400	27JUL89	400	27JUL89	400	27JUL89	400	27JUL89	0
5.4.1	CONDUCT EVALUATIONS 5	99.9	123.0	400	27JUL89	400	27JUL89	500	27DEC89	500	27DEC89	0
6.	CSC INT & TESTING PHASE	0.0	0.0	500	27DEC89	500	27DEC89	500	27DEC89	500	27DEC89	0
6.1	SW DEVELOPMENT TASKS 6	0.0	0.0	500	27DEC89	500	27DEC89	500	27DEC89	500	27DEC89	0
6.1.1	INT & TEST SW	146.5	1394.0	500	27DEC89	500	27DEC89	646	06AUG90	646	06AUG90	0
6.1.2	DEVELOP STD(S)	133.2	410.0	500	27DEC89	513	17JAN90	633	17JUL90	646	06AUG90	13

PROJECT: PMTRADE EFFORT: 410.0 PERSON-MONTHS DURATION: 33.3 MONTHS

CODE	TASK	DUR	EFFORT	START EARLY DAY	START EARLY DATE	START LATE DAY	START LATE DATE	FINISH EARLY DAY	FINISH EARLY DATE	FINISH LATE DAY	FINISH LATE DATE	SLACK TIME DAYS
6.1.3	CONDUCT TRR	20.0	246.0	646	06AUG90	646	06AUG90	666	05SEP90	666	05SEP90	0
6.1.4	TRR COMPLETED	0.0	0.0	666	05SEP90	666	05SEP90	666	05SEP90	666	05SEP90	0
6.2	SW MANAGEMENT TASKS 6	0.0	0.0	500	27DEC89	519	25JAN90	500	27DEC89	519	25JAN90	20
6.2.1	CONDUCT REVIEWS 6	146.5	205.0	500	27DEC89	519	25JAN90	646	06AUG90	666	05SEP90	20
6.3	SCM TASKS 6	0.0	0.0	500	27DEC89	500	27DEC89	500	27DEC89	500	27DEC89	0
6.3.1	OPERATE SW LIBRARY 6	166.5	123.0	500	27DEC89	500	27DEC89	666	05SEP90	666	05SEP90	0
6.4	SQE TASKS 6	0.0	0.0	500	27DEC89	500	27DEC89	500	27DEC89	500	27DEC89	0
6.4.1	CONDUCT EVALUATIONS 6	166.5	164.0	500	27DEC89	500	27DEC89	666	05SEP90	666	05SEP90	0
7.	CSCI TESTING PHASE	0.0	0.0	666	05SEP90	666	05SEP90	666	05SEP90	666	05SEP90	0
7.1	SW DEVELOPMENT TASKS 7	0.0	0.0	666	05SEP90	666	05SEP90	666	05SEP90	666	05SEP90	0
7.1.1	CONDUCT SW T & E	26.6	246.0	666	05SEP90	666	05SEP90	693	16OCT90	693	16OCT90	0
7.1.2	DEVELOP STR(S)	20.0	164.0	693	16OCT90	693	16OCT90	713	14NOV90	713	14NOV90	0
7.1.3	DEVELOP SPS(S)	46.6	164.0	666	05SEP90	666	05SEP90	713	14NOV90	713	14NOV90	0
7.1.4	DEVELOP O&S DOCUMENTS	46.6	164.0	666	05SEP90	666	05SEP90	713	14NOV90	713	14NOV90	0
7.1.5	CONDUCT SW FCA/PCA	20.0	82.0	713	14NOV90	713	14NOV90	733	17DEC90	733	17DEC90	0
7.1.6	SW FCA/PCA COMPLETED	0.0	0.0	733	17DEC90	733	17DEC90	733	17DEC90	733	17DEC90	0
7.2	SW MANAGEMENT TASKS 7	0.0	0.0	666	05SEP90	679	24SEP90	666	05SEP90	679	24SEP90	13
7.2.1	CONDUCT REVIEWS 7	53.3	82.0	666	05SEP90	679	24SEP90	719	26NOV90	733	17DEC90	13
7.3	SCM TASKS 7	0.0	0.0	666	05SEP90	666	05SEP90	666	05SEP90	666	05SEP90	0
7.3.1	OPERATE SW LIBRARY 7	66.6	82.0	666	05SEP90	666	05SEP90	733	17DEC90	733	17DEC90	0
7.4	SQE TASKS 7	0.0	0.0	666	05SEP90	666	05SEP90	666	05SEP90	666	05SEP90	0
7.4.1	CONDUCT EVALUATIONS 7	66.6	82.0	666	05SEP90	666	05SEP90	733	17DEC90	733	17DEC90	0
8.	SYSTEM INT & TEST PHASE	0.0	0.0	733	17DEC90	733	17DEC90	733	17DEC90	733	17DEC90	0
8.1	SW DEVELOPMENT TASKS 8	0.0	0.0	733	17DEC90	733	17DEC90	733	17DEC90	733	17DEC90	0
8.1.1	SUPPORT SYSTEM T&E	133.2	2050.0	733	17DEC90	733	17DEC90	866	08JUL91	866	08JUL91	0
8.1.2	PREPARE PRODUCT BASELINE	33.3	328.0	866	08JUL91	866	08JUL91	899	23AUG91	899	23AUG91	0
8.1.3	SUPPORT SYSTEM FCA/PCA	33.3	82.0	866	08JUL91	866	08JUL91	899	23AUG91	899	23AUG91	0
8.1.4	SYSTEM FCA/PCA COMPLETED	0.0	0.0	899	23AUG91	899	23AUG91	899	23AUG91	899	23AUG91	0
8.2	SW MANAGEMENT TASKS 8	0.0	0.0	733	17DEC90	773	15FEB91	733	17DEC90	773	15FEB91	40
8.2.1	CONDUCT REVIEWS 8	126.5	246.0	733	17DEC90	773	15FEB91	859	25JUN91	899	23AUG91	40
8.3	SCM TASKS 8	0.0	0.0	733	17DEC90	733	17DEC90	733	17DEC90	733	17DEC90	0
8.3.1	OPERATE SW LIBRARY 8	166.5	41.0	733	17DEC90	733	17DEC90	899	23AUG91	899	23AUG91	0
8.4	SQE TASKS 8	0.0	0.0	733	17DEC90	733	17DEC90	733	17DEC90	733	17DEC90	0
8.4.1	CONDUCT EVALUATIONS 8	166.5	82.0	733	17DEC90	733	17DEC90	899	23AUG91	899	23AUG91	0
FINISH		0.0	0.0	899	23AUG91	899	23AUG91	899	23AUG91	899	23AUG91	0

IITRI2.END

****** Tier 1 Distribution ******

Title	Budget	Schedule	Value
Class of Software	1.000	0.950	Ground
Hardware System Type	1.120	1.120	Distributed
Pct of Memory Utilized	0.950	0.990	50 %
S W Configuration Items	1.000	1.000	2
Development Locations	1.000	1.000	1
Customer Locations	1.020	1.020	5
Workstation Types	1.010	1.010	2
Primary Software Language	1.050	1.025	Ada
Pct of Micro-Code	1.030	1.030	5 %
Security Level	1.000	1.000	1

Software Budget Multiplier	1.18547
Software Schedule Multiplier	1.20597
Budget Data Factor	1.000
Schedule Data Factor	1.000

Press any key to continue...

IITRI2.END

Direct Input Mode for SLOC

Software Type	High Order Language New	High Order Language Mod	High Order Language Rehost	Assembly Language New	Assembly Language Mod	Assembly Language Rehost
Systems	0	0	0	0	0	0
Application	32022	1553	3267	0	0	0
Support	10501	1535	0	6545	0	0
Security	0	0	0	0	0	0
Data Statements	42523	3088	3267	6545	0	0
0	Total		48878	Total		6545

New HOL Equivalent	High Order Language New	High Order Language Mod	High Order Language Rehost	Assembly Language New	Assembly Language Mod	Assembly Language Rehost
	42523	2254	327	2182	0	0
Total 47286	Total		45104	Total		2182

Arrow keys, TAB - change input field. F9 - help. INS - save.

IITRI2.END

****** Tier 3 Distribution ******

Title	Budget	Schedule	Complexity
System Requirements	1.000	1.000	Average
Software Requirements	1.000	1.000	Average
Software Documentation	1.050	1.025	Very Complex
Travel Requirements	1.020	1.010	Complex
Man Interaction	1.020	1.010	Complex
Timing and Criticality	1.020	1.010	Complex
Software Testability	1.020	1.020	Complex
Hardware Constraints	0.900	0.990	Simple
Hardware Experience	1.020	1.020	Complex
Software Experience	1.020	1.020	Complex
Software Interfaces	1.000	1.000	Average
Development Facilities	0.950	0.990	Simple
Development vs Host Sys	1.000	1.000	Average
Technology Impacts	1.000	1.000	Average

Press any key to continue...

IITRI1.END

****** Tier 3 Distribution ******

Title	Budget	Schedule	Complexity
COTS Software	1.020	1.020	Complex
Development Team	1.000	1.000	Average
Embedded Development Sys	0.950	0.990	Simple
Software Development Tools	1.020	1.020	Complex
Personnel Resources	1.000	1.000	Average
Programming Language	1.030	1.025	Very Complex

Software Systems Budget Multiplier	1.02924
Software Systems Schedule Multiplier	1.15963

Press any key to continue...

IITRI.END

Summary of Software Development Effort by Organization . Phase
152 ManHours / ManMonth)

Phase	Systems	Software	Test	Quality	Total
Systems Reqts	18.55	32.57	3.84	2.47	57.42
Reqts Allocation	8.35	19.72	2.33	1.49	31.90
Software Reqts	11.88	38.33	4.31	2.90	57.42
Preliminary Design	7.22	31.60	3.45	2.39	44.66
Detailed Design	12.06	70.15	8.18	5.31	95.70
Code	8.84	62.04	9.25	4.70	84.83
Checkout	4.47	32.36	6.39	2.45	45.68
Unit Testing	5.59	39.76	10.36	3.01	58.73
Formal/Phys Qual Test	4.18	29.10	10.19	2.20	45.68
Systems Test & Integ	9.31	70.71	38.60	5.36	123.98
Total	90.44	426.35	96.90	32.30	645.98

ress any key to continue...

IITRI.END

Summary of Software Development Effort by Organization & Review
(152 ManHours / ManMonth)

Review	Systems	Software	Test	Quality	Total
Sys Planning Review	3.37	5.22	0.58	0.40	9.57
Sys Requirements Rev	19.51	37.06	4.42	2.81	63.80
Sys Design Review	14.11	41.76	4.76	3.16	63.80
Preliminary Design Rev	8.97	38.19	4.18	2.89	54.23
Critical Design Review	12.06	70.14	8.19	5.31	95.70
1st Test Readiness Rev	13.33	94.44	15.57	7.15	130.50
2nd Test Readiness Rev	5.59	39.76	10.36	3.01	58.73
Func/Phys Config Audit	5.89	41.04	15.20	3.11	65.25
Acceptance Review	7.59	58.73	33.63	4.45	104.40
Total	90.44	426.35	96.90	32.30	645.98

ress any key to continue...

IITRI.END

Summary of Manloading by Organization(Man Months)

Total Engineering (646.0 MM)

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1987												12.0
1988	12.0	12.2	12.4	12.8	13.2	13.7	14.1	14.6	15.1	15.5	15.9	16.2
1989	16.6	16.8	17.0	17.1	17.2	17.2	17.2	17.1	17.0	16.8	16.5	16.3
1990	16.0	15.6	15.3	14.9	14.6	14.2	13.8	13.5	13.2	12.9	12.6	12.4
1991	12.1	12.0	11.9	11.8	11.7	11.7	11.7	11.8				

Press any key to continue...

IITRI2.END

Summary of Software Development Effort by Organization & Phase
(152 ManHours / ManMonth)

Phase	Systems	Software	Test	Quality	Total
Systems Reqts	16.69	29.28	3.47	2.22	51.67
Reqts Allocation	7.53	17.71	2.12	1.34	28.70
Software Reqts	10.69	34.45	3.92	2.61	51.67
Preliminary Design	6.48	28.42	3.13	2.15	40.19
Detailed Design	10.79	63.13	7.41	4.78	86.11
Code	7.70	54.62	8.17	4.14	74.63
Checkout	3.89	28.50	5.64	2.16	40.19
Unit Testing	4.86	35.01	9.14	2.65	51.67
Formal/Phys Qual Test	3.63	25.61	9.00	1.94	40.19
Systems Test & Integ	8.10	62.16	34.11	4.71	109.07
Total	80.37	378.89	86.11	28.70	574.08

Press any key to continue...

IITRI2.END

Summary of Software Development Effort by Organization & Review
(152 ManHours / ManMonth)

Review	Systems	Software	Test	Quality	Total
Sys Planning Review	3.03	4.70	0.53	0.36	8.61
Sys Requirements Rev	17.58	33.30	4.00	2.52	57.41
Sys Design Review	12.71	37.52	4.33	2.84	57.41
Preliminary Design Rev	8.05	34.35	3.79	2.60	48.80
Critical Design Review	10.79	63.12	7.41	4.78	86.11
1st Test Readiness Rev	11.61	83.15	13.76	6.30	114.82
2nd Test Readiness Rev	4.86	35.00	9.15	2.65	51.67
Func/Phys Config Audit	5.13	36.12	13.42	2.74	57.41
Acceptance Review	6.60	51.61	29.72	3.91	91.85
Total	80.37	378.89	86.11	28.70	574.08

Press any key to continue...

IITRI2.END

Summary of Manloading by Organization(Man Months)

Total Engineering (574.1 MM)

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1987												17.1
1988	17.3	17.9	18.8	19.9	21.0	22.0	22.9	23.7	24.2	24.5	24.6	24.5
1989	24.2	23.7	23.1	.22.4	21.6	20.7	19.9	19.1	18.4	17.8	17.3	17.0
1990	16.8	16.7	16.8									

Press any key to continue...

IITRI.END

Summary of Software Schedule by Phase
(Calendar Months)

Phase	(CDR Specified) User Schedule Month/Year From	To	(CDR Not Specified) Derived Schedule Month/Year From	To
Systems Requirements	12/1987 -	4/1988	12/1987 -	2/1988
Requirements Allocation	4/1988 -	7/1988	2/1988 -	4/1988
Software Requirements	7/1988 -	11/1988	4/1988 -	6/1988
Preliminary Design	8/1988 -	2/1989	5/1988 -	8/1988
Detailed Design	2/1989 -	7/1989	8/1988 -	12/1988
Code	6/1989 -	4/1990	11/1988 -	5/1989
Checkout	10/1989 -	8/1990	1/1989 -	7/1989
Unit Testing	1/1990 -	10/1990	4/1989 -	9/1989
PQT / FQT, Integration	4/1990 -	1/1991	5/1989 -	11/1989
Systems Test & Integration	1/1991 -	8/1991	11/1989 -	3/1990

Press any key to continue...

IITRI.END

Summary of Software Schedule by Review
(Calendar Months)

Review	(CDR Specified) User Schedule Month/Year	(CDR Not Specified) Derived Schedule Month/Year
Systems Planning Review	12/1987	12/1987
Systems Requirements Review	4/1988	2/1988
System Design Review	9/1988	5/1988
Preliminary Design Review	12/1988	7/1988
Critical Design Review	7/1989	12/1988
1st Test Readiness Review	4/1990	5/1989
2nd Test Readiness Review	8/1990	8/1989
Func/Phys Configuration Audits	1/1991	11/1989
Acceptance Review	8/1991	3/1990

Press any key to continue...

APPENDIX G

ADAMAT REPORT

```
Date:  12-30-1991                                    page:  1
DUA0:[USER.ADAMAT.UH1_DATA]APPLICATION.REP_COM;1
Score    Good    Total  |Level------- Metric Name

 0.44   43650    98225  |1----------- RELIABILITY
 0.54   87611   161623  |1----------- MAINTAINABILITY
 0.96  487577   505938  |1----------- PORTABILITY
 0.86  548183   635673  |1----------- ALL_CRITERIA
                 41354  | 2---------- SLOC
                189246  |  3--------- PHYSICAL_LINES
                 63800  |   4-------- PHYSICAL_ADA_LINES
                 62528  |    5------- ADA_UNCOMMENTED_LINES
                  1272  |    5------- ADA_COMMENTED_LINES
                  1270  |     6------ COMMENTED_LINES_WITH_TEXT
                     2  |     6------ COMMENTED_LINES_BLANK
                107530  |   4-------- PHYSICAL_COMMENT_LINES
                106956  |    5------- COMMENT_LINES_WITH_TEXT
                   574  |    5------- COMMENT_LINES_BLANK
                 17916  |   4-------- PHYSICAL_BLANK_LINES
                 41354  |  3--------- LOGICAL_LINES
                 37267  |   4-------- STATEMENTS
                 24129  |    5------- EXECUTABLE_STATEMENTS
                 13138  |    5------- DECLARATIVE_STATEMENTS
                  3627  |   4-------- CONTEXT_CLAUSES
                  2241  |    5------- WITH_CLAUSES
                  1386  |    5------- USE_CLAUSES
                   460  |   4-------- PRAGMAS
 0.65   11578    17789  | 2---------- ANOMALY_MANAGEMENT
 0.39    2887     7345  |  3--------- PREVENTION
 0.31    1331     4252  |   4-------- APPLICATIVE_DECLARATIONS
 0.50     444      891  |    5------- APPLICATIVE_DECL_SPECIFICATION
 0.26     887     3361  |    5------- APPLICATIVE_DECL_BODY
 0.48    1392     2882  |   4-------- DEFAULT_INITIALIZATION
 0.30     129      428  |    5------- DEFAULT_INIT_SPECIFICATION
 0.51    1263     2454  |    5------- DEFAULT_INIT_BODY
 0.71      77      108  |   4-------- NORMAL_LOOPS
            0        0  |   4-------- CONSTRAINED_NUMERICS
 0.92      87       95  |   4-------- CONSTRAINED_SUBTYPE
 0.00       0        8  |   4-------- CONSTRAINED_VARIANT_RECORDS
 0.83    8676    10399  |  3--------- DETECTION
            0        0  |   4-------- SUPPRESS_PRAGMA
                     0  |    5------- CONSTRAINT_ERROR
                     0  |    5------- PROGRAM_ERROR
                     0  |    5------- STORAGE_ERROR
                     0  |    5------- NUMERIC_ERROR
 0.83    8676    10399  |   4-------- USER_TYPES
 0.90     903      998  |    5------- USER_TYPES_FOR_PARAMETERS
 0.82    3740     4535  |    5------- USER_TYPES_SPECIFICATION
 0.83    4033     4866  |    5------- USER_TYPES_BODY
 0.33      15       45  |  3--------- RECOVERY
 0.33      15       45  |   4-------- USER_EXCEPTIONS_RAISED
 0.98  448994   456261  | 2---------- INDEPENDENCE
 0.97  162890   167974  |  3--------- IO_INDEP
                     7  |   4-------- NO_MISSED_CLOSE
 0.97  104729   107567  |   4-------- NO_SYS_DEP_IO
 0.96   58161    60400  |   4-------- IO_NON_MIX
 0.98   37737    38333  |  3--------- TASK_INDEP
 0.99   33569    33867  |   4-------- NO_TASK_STMT
 0.93    4168     4466  |   4-------- TASK_STMT_NON_MIX
 0.99  139968   141366  |  3--------- MACH_INDEP
```

```
Date:  12-30-1991                                         page:  2
DUA0:[USER.ADAMAT.UH1_DATA]APPLICATION.REP_COM;1
Score    Good    Total |Level------- Metric Name
```

Score	Good	Total	Level------- Metric Name
0.00	0	7	4-------- MACHARITHINDEP
	0	0	5------- PACKAGE_ARITH_INDEP
		0	6------ NO_MAX_INT
		0	6------ NO_MIN_INT
		0	6------ NO_MAX_DIGITS
		0	6------ NO_MAX_MANTISSA
		0	6------ NO_FINE_DELTA
		0	6------ NO_TICK
		4	5------- NO_INTEGER_DECL
		0	5------- NO_SHORT_INTEGER_DECL
		0	5------- NO_LONG_INTEGER_DECL
		0	5------- NO_FLOAT_DECL
		0	5------- NO_SHORT_FLOAT_DECL
		0	5------- NO_LONG_FLOAT_DECL
		1	5------- NO_NATURAL_DECL
		2	5------- NO_POSITIVE_DECL
	0	0	5------- FIXED_CLAUSE
0.60	1615	2706	4-------- MACHREPINDEP
		11	5------- NO_PRAGMA_PACK
0.14	173	1219	5------- NUMERIC_CONSTANT_DECL
	0	0	5------- NUMERIC_TYPE_DECLARATIONS
0.98	1442	1476	5------- CLAUSE_REP_INDEP
1.00	833	833	6------ NO_LENGTH_CLAUSE_FOR_SIZE
0.76	25	33	6------ NO_LENGTH_CLAUSE_FOR_STORAGE_SIZE
0.96	292	305	6------ NO_ALIGNMENT_CLAUSE_FOR_RECORD_TYPES
0.96	292	305	6------ NO_COMPONENT_CLAUSE_FOR_RECORD_TYPES
0.99	112368	112668	4-------- MACHCONFIGINDEP
0.99	5043	5101	5------- NO_ADDRESS_CLAUSE_IN_DECL
		0	5------- NO_PRAG_SYS_PARAM
0.99	107325	107567	5------- NO_REP_ATTRIBUTE
1.00	25985	25985	4-------- MACHCODEINDEP
1.00	25985	25985	5------- NO_MACH_CODE_STMT
0.99	108399	108410	3--------- SOFT_INDEP
1.00	107567	107567	4-------- NO_SYS_DEP_MOD
		4	4-------- NO_IMPL_DEP_PRAGMAS
		6	4-------- NO_PRAGMA_INTERFACE
0.99	832	833	4-------- NON_ACCESS_TYPE
		0	4-------- NO_IMPL_DEP_ATTRS
0.00	0	178	3--------- PHYS_LIM_INDEP
		178	4-------- COMPILER_LIMIT
0.68	5490	8123	2---------- MODULARITY
0.62	3616	5834	3--------- INFORMATION_HIDING
0.69	3616	5212	4-------- HIDDEN_INFORMATION
0.67	887	1331	5------- CONSTANTS_HID
	0	0	5------- EXCEPTIONS_HID
0.85	2474	2921	5------- VARIABLES_HID
0.25	211	833	5------- TYPES_HID
0.46	44	95	5------- SUBTYPES_HID
0.00	0	32	5------- TASKS_HID
0.00	0	622	4-------- PRIVATE_INFORMATION
0.00	0	622	5------- PRIVATE_TYPES
	0	0	5------- LIMITED_PRIVATE_TYPES
	0	0	5------- PRIVATE_TYPE_AND_PART
	0	0	5------- PRIVATE_TYPE_AND_CONSTANT
0.95	1537	1611	3--------- PROFILE
0.95	1536	1610	4-------- LIMITED_SIZE_PROFILE

Date: 12-30-1991 page: 3
DUA0:[USER.ADAMAT.UH1_DATA]APPLICATION.REP_COM;1

Score	Good	Total	Level------- Metric Name
1.00	1	1	4-------- SIMPLE_BLOCKS
0.50	337	678	3--------- COUPLING
0.68	232	339	4-------- NO_MULTIPLE_TYPE_DECLARATIONS
0.31	105	339	4-------- NO_VARIABLE_DECLARATIONS_IN_SPEC
0.80	33093	41554	2---------- SELF_DESCRIPTIVENESS
0.70	18974	27208	3--------- COMMENTS
0.81	13108	16263	4-------- N_COMMENTS
1.00	2917	2917	5------- NCS_SPEC
1.00	1695	1695	6------ NCS_PACKAGE_SPEC
1.00	160	160	6------ NCS_TASK_SPEC
1.00	1062	1062	6------ NCS_SUBPROG_SPEC
0.99	4173	4178	5------- NCS_BODY
1.00	740	740	6------ NCS_PACKAGE_BODY
1.00	160	160	6------ NCS_TASK_BODY
1.00	3273	3273	6------ NCS_SUBPROG_BODY
0.00	0	5	6------ NCS_SUBUNIT
	0	0	6------ NCS_BODY_STUB
0.31	1254	4109	5------- NCS_STATEMENTS
0.02	3	155	6------ NCS_EXIT
0.07	12	182	6------ NCS_RETURN
	0	0	6------ NCS_GOTO
	0	0	6------ NCS_ABORT
0.82	119	145	6------ NCS_DELAY
	0	0	6------ NCS_TERMINATE
0.01	31	2241	6------ NCS_WITH
0.79	1089	1386	6------ NCS_USE
0.94	4764	5059	5------- NCS_DECLARATIONS
0.80	367	460	6------ NCS_PRAGMA
0.00	0	39	6------ NCS_RECORD_REPRESENTATION
0.50	87	174	6------ NCS_ADDRESS_CLAUSE
0.51	20	39	6------ NCS_ALIGNMENT_CLAUSE
1.00	24	24	6------ NCS_LENGTH_CLAUSE
1.00	1331	1331	6------ NCS_CONSTANT_DECL
1.00	2912	2912	6------ NCS_VARIABLE_DECL
0.29	23	80	6------ NCS_ENTRY_DECL
0.54	5866	10945	4-------- N_COMMENTED
0.83	605	729	5------- NCO_SPEC
0.98	333	339	6------ NCO_PACKAGE_SPEC
1.00	36	36	6------ NCO_TASK_SPEC
0.67	236	354	6------ NCO_SUBPROG_SPEC
0.88	1122	1272	5------- NCO_BODY
0.97	144	148	6------ NCO_PACKAGE_BODY
1.00	32	32	6------ NCO_TASK_BODY
0.87	946	1091	6------ NCO_SUBPROG_BODY
0.00	0	1	6------ NCO_SUBUNIT
	0	0	6------ NCO_BODY_STUB
0.09	362	4109	5------- NCO_STATEMENTS
0.01	2	155	6------ NCO_EXIT
0.03	6	182	6------ NCO_RETURN
	0	0	6------ NCO_GOTO
	0	0	6------ NCO_ABORT
0.19	28	145	6------ NCO_DELAY
	0	0	6------ NCO_TERMINATE
0.01	20	2241	6------ NCO_WITH
0.22	306	1386	6------ NCO_USE
0.78	3777	4835	5------- NCO_DECLARATIONS

Date: 12-30-1991 page: 4
DUA0:[USER.ADAMAT.UH1_DATA]APPLICATION.REP_COM;1

Score	Good	Total	Level------- Metric Name
0.26	118	460	6------ NCO_PRAGMA
0.00	0	13	6------ NCO_RECORD_REPRESENTATION
0.59	34	58	6------ NCO_ADDRESS_CLAUSE
0.69	9	13	6------ NCO_ALIGNMENT_CLAUSE
1.00	8	8	6------ NCO_LENGTH_CLAUSE
0.95	1258	1331	6------ NCO_CONSTANT_DECL
0.80	2329	2912	6------ NCO_VARIABLE_DECL
0.52	21	40	6------ NCO_ENTRY_DECL
0.98	14119	14346	3--------- IDENTIFIER
0.98	14119	14346	4-------- NO_PREDEFINED_WORDS
0.40	32072	80436	2---------- SIMPLICITY
0.83	3424	4116	3--------- CODING_SIMPLICITY
0.84	3140	3725	4-------- SIMPLE_BOOLEAN_EXPRESSION
0.73	284	391	4-------- EXPRES_TO_DO_BOOLEAN_ASSIGN
0.13	4985	39502	3--------- DESIGN_SIMPLICITY
0.16	954	6124	4-------- CALLS_TO_PROCEDURES
0.37	150	403	4-------- ARRAY_TYPE_EXPLICIT
0.17	95	557	4-------- SUBTYPE_EXPLICIT
0.72	84	116	4-------- ARRAY_RANGE_TYPE_EXPLICIT
0.11	3702	32302	4-------- DECLARATIONS_CONTAIN_LITERALS
0.64	23663	36818	3--------- FLOW_SIMPLICITY
0.49	2135	4321	4-------- BRANCH_CONSTRUCTS
0.96	1052	1091	4-------- SINGLE_EXIT_SUBPROGRAM
0.64	413	649	4-------- FOR_LOOPS
0.97	1563	1610	4-------- LEVEL_OF_NESTING_BY_MODULE
0.83	1504	1815	4-------- LEVEL_OF_NESTING
0.95	4093	4321	4-------- STRUCTURED_BRANCH_CONSTRUCT
0.85	3672	4321	4-------- NON_BACK_BRANCH_CONSTRUCT
	0	0	4-------- NO_LABELS
0.46	2007	4377	4-------- DECISIONS
	0	0	4-------- GOTOS
0.39	4321	11093	4-------- BRANCH_AND_NESTING
0.91	1460	1610	4-------- CYCLOMATIC_COMPLEXITY
0.90	1443	1610	4-------- MULTIPLE_COND_CYCLOMATIC_COMPLEXITY
0.54	16956	31510	2---------- SYSTEM_CLARITY
0.54	16956	31510	3--------- STYLE
0.60	5451	9131	4-------- EXPRESSION_STYLE
0.87	3225	3725	5------- NON_NEGATED_BOOLEAN_EXPRESSIONS
0.37	1605	4344	5------- EXPRESSIONS_PARENTHESIZED
0.80	521	649	5------- NO_WHILE_LOOPS
0.24	100	413	5------- FOR_LOOPS_WITH_TYPE
0.88	6197	7028	4-------- DECLARATION_STYLE
0.16	156	975	5------- NO_DEFAULT_MODE_PARAMETERS
	0	0	5------- PRIVATE_ACCESS_TYPES
0.99	5788	5800	5------- SINGLE_OBJECT_DECLARATION_LISTS
1.00	253	253	5------- SINGLE_IMPLICIT_TYPE_ARRAY
	0	0	5------- NO_INITIALIZATION_BY_NEW
0.61	3733	6162	4-------- NAMING_STYLE
0.00	1	650	5------- STRUCTURES_NAMED
0.00	1	649	6------ NAMED_LOOPS
0.00	0	1	6------ NAMED_BLOCKS
0.72	1643	2292	5------- STRUCTURE_ENDS_WITH_NAME
1.00	1642	1642	6------ MODULE_END_WITH_NAME
0.00	1	649	6------ LOOP_END_WITH_NAME
0.00	0	1	6------ BLOCK_END_WITH_NAME
0.01	1	155	5------- NAMED_EXITS

ADAMAT Results

Date: 12-30-1991 page: 5
DUA0:[USER.ADAMAT.UH1_DATA]APPLICATION.REP_COM;1

Score	Good	Total	Level------- Metric Name
0.68	2088	3065	5------- NAMED_AGGREGATE
0.17	1575	9189	4-------- QUALIFICATION_STYLE
0.00	0	3065	5------- QUALIFIED_AGGREGATE
0.26	1575	6124	5------- QUALIFIED_SUBPROGRAM

Date: 12-31-1991 page: 1

DUA0:[USER.ADAMAT.UH1_DATA]SUPPORT.REP_COM;1

Score	Good	Total	Level-------	Metric Name
0.48	12905	26875	1-----------	RELIABILITY
0.54	22850	42392	1-----------	MAINTAINABILITY
0.95	107745	113276	1-----------	PORTABILITY
0.84	124568	148136	1-----------	ALL_CRITERIA
		10799	2----------	SLOC
		46732	3---------	PHYSICAL_LINES
		16173	4--------	PHYSICAL_ADA_LINES
		15918	5-------	ADA_UNCOMMENTED_LINES
		255	5-------	ADA_COMMENTED_LINES
		254	6------	COMMENTED_LINES_WITH_TEXT
		1	6------	COMMENTED_LINES_BLANK
		24803	4--------	PHYSICAL_COMMENT_LINES
		24652	5-------	COMMENT_LINES_WITH_TEXT
		151	5-------	COMMENT_LINES_BLANK
		5756	4--------	PHYSICAL_BLANK_LINES
		10799	3---------	LOGICAL_LINES
		9958	4--------	STATEMENTS
		6931	5-------	EXECUTABLE_STATEMENTS
		3027	5-------	DECLARATIVE_STATEMENTS
		830	4--------	CONTEXT_CLAUSES
		488	5-------	WITH_CLAUSES
		342	5-------	USE_CLAUSES
		11	4--------	PRAGMAS
0.67	2902	4333	2----------	ANOMALY_MANAGEMENT
0.44	825	1883	3---------	PREVENTION
0.46	559	1207	4--------	APPLICATIVE_DECLARATIONS
0.69	406	587	5-------	APPLICATIVE_DECL_SPECIFICATION
0.25	153	620	5-------	APPLICATIVE_DECL_BODY
0.37	236	632	4--------	DEFAULT_INITIALIZATION
0.27	48	179	5-------	DEFAULT_INIT_SPECIFICATION
0.42	188	453	5-------	DEFAULT_INIT_BODY
0.45	10	22	4--------	NORMAL_LOOPS
	0	0	4--------	CONSTRAINED_NUMERICS
1.00	20	20	4--------	CONSTRAINED_SUBTYPE
0.00	0	2	4--------	CONSTRAINED_VARIANT_RECORDS
0.85	2070	2443	3---------	DETECTION
	0	0	4--------	SUPPRESS_PRAGMA
		0	5-------	CONSTRAINT_ERROR
		0	5-------	PROGRAM_ERROR
		0	5-------	STORAGE_ERROR
		0	5-------	NUMERIC_ERROR
0.85	2070	2443	4--------	USER_TYPES
0.86	128	148	5-------	USER_TYPES_FOR_PARAMETERS
0.89	1427	1604	5-------	USER_TYPES_SPECIFICATION
0.75	515	691	5-------	USER_TYPES_BODY
1.00	7	7	3---------	RECOVERY
1.00	7	7	4--------	USER_EXCEPTIONS_RAISED
0.97	98816	101411	2----------	INDEPENDENCE
0.95	33762	35428	3---------	IO_INDEP
		0	4--------	NO_MISSED_CLOSE
0.96	21154	22048	4--------	NO_SYS_DEP_IO
0.94	12608	13380	4--------	IO_NON_MIX
0.97	11719	12115	3---------	TASK_INDEP
0.98	9216	9414	4--------	NO_TASK_STMT
0.93	2503	2701	4--------	TASK_STMT_NON_MIX
0.98	31122	31629	3---------	MACH_INDEP

Date: 12-31-1991 page: 2
DUA0:[USER.ADAMAT.UH1_DATA]SUPPORT.REP_COM;1

Score	Good	Total	Level------- Metric Name
	0	0	4-------- MACHARITHINDEP
	0	0	5------- PACKAGE_ARITH_INDEP
		0	6------ NO_MAX_INT
		0	6------ NO_MIN_INT
		0	6------ NO_MAX_DIGITS
		0	6------ NO_MAX_MANTISSA
		0	6------ NO_FINE_DELTA
		0	6------ NO_TICK
		0	5------- NO_INTEGER_DECL
		0	5------- NO_SHORT_INTEGER_DECL
		0	5------- NO_LONG_INTEGER_DECL
		0	5------- NO_FLOAT_DECL
		0	5------- NO_SHORT_FLOAT_DECL
		0	5------- NO_LONG_FLOAT_DECL
		0	5------- NO_NATURAL_DECL
		0	5------- NO_POSITIVE_DECL
	0	0	5------- FIXED_CLAUSE
0.46	381	831	4-------- MACHREPINDEP
		7	5------- NO_PRAGMA_PACK
0.16	80	509	5------- NUMERIC_CONSTANT_DECL
	0	0	5------- NUMERIC_TYPE_DECLARATIONS
0.96	301	315	5------- CLAUSE_REP_INDEP
1.00	166	166	6------ NO_LENGTH_CLAUSE_FOR_SIZE
1.00	15	15	6------ NO_LENGTH_CLAUSE_FOR_STORAGE_SIZE
0.90	60	67	6------ NO_ALIGNMENT_CLAUSE_FOR_RECORD_TYPES
0.90	60	67	6------ NO_COMPONENT_CLAUSE_FOR_RECORD_TYPES
0.99	23310	23367	4-------- MACHCONFIGINDEP
0.98	1287	1319	5------- NO_ADDRESS_CLAUSE_IN_DECL
		0	5------- NO_PRAG_SYS_PARAM
0.99	22023	22048	5------- NO_REP_ATTRIBUTE
1.00	7431	7431	4-------- MACHCODEINDEP
1.00	7431	7431	5------- NO_MACH_CODE_STMT
0.99	22213	22217	3--------- SOFT_INDEP
1.00	22048	22048	4-------- NO_SYS_DEP_MOD
		1	4-------- NO_IMPL_DEP_PRAGMAS
		2	4-------- NO_PRAGMA_INTERFACE
0.99	165	166	4-------- NON_ACCESS_TYPE
		0	4-------- NO_IMPL_DEP_ATTRS
0.00	0	22	3--------- PHYS_LIM_INDEP
		22	4-------- COMPILER_LIMIT
0.50	971	1955	2---------- MODULARITY
0.43	658	1537	3--------- INFORMATION_HIDING
0.47	658	1407	4-------- HIDLEN_INFORMATION
0.27	153	559	5------- CONSTANTS_HID
	0	0	5------- EXCEPTIONS_HID
0.72	467	648	5------- VARIABLES_HID
0.22	36	166	5------- TYPES_HID
0.10	2	20	5------- SUBTYPES_HID
0.00	0	14	5------- TASKS_HID
0.00	0	130	4-------- PRIVATE_INFORMATION
0.00	0	130	5------- PRIVATE_TYPES
	0	0	5------- LIMITED_PRIVATE_TYPES
	0	0	5------- PRIVATE_TYPE_AND_PART
	0	0	5------- PRIVATE_TYPE_AND_CONSTANT
0.91	242	266	3--------- PROFILE
0.91	242	266	4-------- LIMITED_SIZE_PROFILE

Date: 12-31-1991 page: 3
DUA0:[USER.ADAMAT.UH1_DATA]SUPPORT.REP_COM;1

Score	Good	Total	Level------- Metric Name
	0	0	4-------- SIMPLE_BLOCKS
0.47	71	152	3--------- COUPLING
0.68	52	76	4-------- NO_MULTIPLE_TYPE_DECLARATIONS
0.25	19	76	4-------- NO_VARIABLE_DECLARATIONS_IN_SPEC
0.80	7958	9910	2---------- SELF_DESCRIPTIVENESS
0.71	4586	6478	3--------- COMMENTS
0.78	2952	3774	4-------- N_COMMENTS
1.00	627	627	5------- NCS_SPEC
1.00	380	380	6------ NCS_PACKAGE_SPEC
1.00	70	70	6------ NCS_TASK_SPEC
1.00	177	177	6------ NCS_SUBPROG_SPEC
0.99	674	679	5------- NCS_BODY
1.00	190	190	6------ NCS_PACKAGE_BODY
1.00	70	70	6------ NCS_TASK_BODY
1.00	414	414	6------ NCS_SUBPROG_BODY
0.00	0	5	6------ NCS_SUBUNIT
	0	0	6------ NCS_BODY_STUB
0.35	381	1083	5------- NCS_STATEMENTS
0.11	3	27	6------ NCS_EXIT
0.13	12	91	6------ NCS_RETURN
	0	0	6------ NCS_GOTO
	0	0	6------ NCS_ABORT
0.83	112	135	6------ NCS_DELAY
	0	0	6------ NCS_TERMINATE
0.06	28	488	6------ NCS_WITH
0.66	226	342	6------ NCS_USE
0.92	1270	1385	5------- NCS_DECLARATIONS
0.82	9	11	6------ NCS_PRAGMA
0.00	0	21	6------ NCS_RECORD_REPRESENTATION
0.54	52	96	6------ NCS_ADDRESS_CLAUSE
0.24	5	21	6------ NCS_ALIGNMENT_CLAUSE
	0	0	6------ NCS_LENGTH_CLAUSE
1.00	559	559	6------ NCS_CONSTANT_DECL
1.00	645	645	6------ NCS_VARIABLE_DECL
0.00	0	32	6------ NCS_ENTRY_DECL
0.60	1634	2704	4-------- N_COMMENTED
0.99	151	153	5------- NCO_SPEC
0.99	75	76	6------ NCO_PACKAGE_SPEC
1.00	18	18	6------ NCO_TASK_SPEC
0.98	58	59	6------ NCO_SUBPROG_SPEC
0.96	184	191	5------- NCO_BODY
0.97	37	38	6------ NCO_PACKAGE_BODY
1.00	14	14	6------ NCO_TASK_BODY
0.96	133	138	6------ NCO_SUBPROG_BODY
0.00	0	1	6------ NCO_SUBUNIT
	0	0	6------ NCO_BODY_STUB
0.08	91	1083	5------- NCO_STATEMENTS
0.07	2	27	6------ NCO_EXIT
0.07	6	91	6------ NCO_RETURN
	0	0	6------ NCO_GOTO
	0	0	6------ NCO_ABORT
0.16	21	135	6------ NCO_DELAY
	0	0	6------ NCO_TERMINATE
0.04	20	488	6------ NCO_WITH
0.12	42	342	6------ NCO_USE
0.95	1208	1277	5------- NCO_DECLARATIONS

Date: 12-31-1991 page: 4
DUA0:[USER.ADAMAT.UH1_DATA]SUPPORT.REP_COM;1

Score	Good	Total	Level------- Metric Name
0.64	7	11	6------ NCO_PRAGMA
0.00	0	7	6------ NCO_RECORD_REPRESENTATION
0.41	13	32	6------ NCO_ADDRESS_CLAUSE
0.71	5	7	6------ NCO_ALIGNMENT_CLAUSE
	0	0	6------ NCO_LENGTH_CLAUSE
0.99	557	559	6------ NCO_CONSTANT_DECL
0.97	626	645	6------ NCO_VARIABLE_DECL
0.00	0	16	6------ NCO_ENTRY_DECL
0.98	3372	3432	3--------- IDENTIFIER
0.98	3372	3432	4-------- NO_PREDEFINED_WORDS
0.44	10003	22542	2---------- SIMPLICITY
0.82	1128	1372	3--------- CODING_SIMPLICITY
0.83	1098	1319	4-------- SIMPLE_BOOLEAN_EXPRESSION
0.57	30	53	4-------- EXPRES_TO_DO_BOOLEAN_ASSIGN
0.30	2731	9048	3--------- DESIGN_SIMPLICITY
0.13	218	1702	4-------- CALLS_TO_PROCEDURES
0.39	35	89	4-------- ARRAY_TYPE_EXPLICIT
0.11	20	178	4-------- SUBTYPE_EXPLICIT
0.00	0	8	4-------- ARRAY_RANGE_TYPE_EXPLICIT
0.35	2458	7071	4-------- DECLARATIONS_CONTAIN_LITERALS
0.51	6144	12122	3--------- FLOW_SIMPLICITY
0.23	353	1535	4-------- BRANCH_CONSTRUCTS
0.81	112	138	4-------- SINGLE_EXIT_SUBPROGRAM
0.68	193	282	4-------- FOR_LOOPS
0.95	252	266	4-------- LEVEL_OF_NESTING_BY_MODULE
0.74	275	371	4-------- LEVEL_OF_NESTING
0.94	1436	1535	4-------- STRUCTURED_BRANCH_CONSTRUCT
0.82	1253	1535	4-------- NON_BACK_BRANCH_CONSTRUCT
	0	0	4-------- NO_LABELS
0.20	318	1580	4-------- DECISIONS
	0	0	4-------- GOTOS
0.35	1535	4348	4-------- BRANCH_AND_NESTING
0.79	211	266	4-------- CYCLOMATIC_COMPLEXITY
0.77	206	266	4-------- MULTIPLE_COND_CYCLOMATIC_COMPLEXITY
0.49	3918	7985	2---------- SYSTEM_CLARITY
0.49	3918	7985	3--------- STYLE
0.59	1797	3036	4-------- EXPRESSION_STYLE
0.89	1169	1319	5------- NON_NEGATED_BOOLEAN_EXPRESSIONS
0.29	358	1242	5------- EXPRESSIONS_PARENTHESIZED
0.76	215	282	5------- NO_WHILE_LOOPS
0.28	55	193	5------- FOR_LOOPS_WITH_TYPE
0.91	1505	1646	4-------- DECLARATION_STYLE
0.09	13	151	5------- NO_DEFAULT_MODE_PARAMETERS
	0	0	5------- PRIVATE_ACCESS_TYPES
0.99	1438	1441	5------- SINGLE_OBJECT_DECLARATION_LISTS
1.00	54	54	5------- SINGLE_IMPLICIT_TYPE_ARRAY
	0	0	5------- NO_INITIALIZATION_BY_NEW
0.32	401	1236	4-------- NAMING_STYLE
0.00	1	282	5------- STRUCTURES_NAMED
0.00	1	282	6------ NAMED_LOOPS
	0	0	6------ NAMED_BLOCKS
0.50	281	562	5------- STRUCTURE_ENDS_WITH_NAME
1.00	280	280	6------ MODULE_END_WITH_NAME
0.00	1	282	6------ LOOP_END_WITH_NAME
	0	0	6------ BLOCK_END_WITH_NAME
0.04	1	27	5------- NAMED_EXITS

```
Date:  12-31-1991                                          page:  5
DUA0:[USER.ADAMAT.UH1_DATA]SUPPORT.REP_COM;1
Score    Good    Total |Level------- Metric Name

 0.32     118      365 |    5------- NAMED_AGGREGATE
 0.10     215     2067 |   4-------- QUALIFICATION_STYLE
 0.00       0      365 |    5------- QUALIFIED_AGGREGATE
 0.13     215     1702 |    5------- QUALIFIED_SUBPROGRAM
```

```
Date:  12-30-1991                                        page:  1
DUA0:[USER.ADAMAT.UH1_DATA]GAT_AND_SERVICES.REP_COM;1
Score    Good    Total  |Level------- Metric Name

 0.57    9354    16455  |1----------- RELIABILITY
 0.69   18312    26453  |1----------- MAINTAINABILITY
 0.97   77892    80314  |1----------- PORTABILITY
 0.90   91112   101721  |1----------- ALL_CRITERIA
                  7256  | 2---------- SLOC
                 50999  |  3--------- PHYSICAL_LINES
                 11841  |   4-------- PHYSICAL_ADA_LINES
                 10574  |    5------- ADA_UNCOMMENTED_LINES
                  1267  |    5------- ADA_COMMENTED_LINES
                  1266  |     6------ COMMENTED_LINES_WITH_TEXT
                     1  |     6------ COMMENTED_LINES_BLANK
                 32029  |   4-------- PHYSICAL_COMMENT_LINES
                 31824  |    5------- COMMENT_LINES_WITH_TEXT
                   205  |    5------- COMMENT_LINES_BLANK
                  7129  |   4-------- PHYSICAL_BLANK_LINES
                  7256  |  3--------- LOGICAL_LINES
                  7007  |   4-------- STATEMENTS
                  4202  |    5------- EXECUTABLE_STATEMENTS
                  2805  |    5------- DECLARATIVE_STATEMENTS
                   228  |   4-------- CONTEXT_CLAUSES
                   140  |    5------- WITH_CLAUSES
                    88  |    5------- USE_CLAUSES
                    21  |   4-------- PRAGMAS
 0.65    2774     4276  | 2---------- ANOMALY_MANAGEMENT
 0.33     505     1525  |  3--------- PREVENTION
 0.17     136      782  |   4-------- APPLICATIVE_DECLARATIONS
 0.46      43       94  |    5------- APPLICATIVE_DECL_SPECIFICATION
 0.14      93      688  |    5------- APPLICATIVE_DECL_BODY
 0.44     281      642  |   4-------- DEFAULT_INITIALIZATION
 0.57      29       51  |    5------- DEFAULT_INIT_SPECIFICATION
 0.43     252      591  |    5------- DEFAULT_INIT_BODY
 0.76      13       17  |   4-------- NORMAL_LOOPS
            0        0  |   4-------- CONSTRAINED_NUMERICS
 0.89      75       84  |   4-------- CONSTRAINED_SUBTYPE
            0        0  |   4-------- CONSTRAINED_VARIANT_RECORDS
 0.86    2185     2545  |  3--------- DETECTION
            0        0  |   4-------- SUPPRESS_PRAGMA
                     0  |    5------- CONSTRAINT_ERROR
                     0  |    5------- PROGRAM_ERROR
                     0  |    5------- STORAGE_ERROR
                     0  |    5------- NUMERIC_ERROR
 0.86    2185     2545  |   4-------- USER_TYPES
 0.89     494      553  |    5------- USER_TYPES_FOR_PARAMETERS
 0.85     869     1022  |    5------- USER_TYPES_SPECIFICATION
 0.85     822      970  |    5------- USER_TYPES_BODY
 0.41      84      206  |  3--------- RECOVERY
 0.41      84      206  |   4-------- USER_EXCEPTIONS_RAISED
 0.99   70026    70992  | 2---------- INDEPENDENCE
 0.98   24922    25428  |  3--------- IO_INDEP
                    13  |   4-------- NO_MISSED_CLOSE
 0.98   16057    16464  |   4-------- NO_SYS_DEP_IO
 0.99    8865     8951  |   4-------- IO_NON_MIX
 0.99    6231     6275  |  3--------- TASK_INDEP
 0.99    6144     6166  |   4-------- NO_TASK_STMT
 0.80      87      109  |   4-------- TASK_STMT_NON_MIX
 0.98   22206    22602  |  3--------- MACH_INDEP
```

Date: 12-30-1991 page: 2
DUA0:[USER.ADAMAT.UH1_DATA]GAT_AND_SERVICES.REP_COM;1

Score	Good	Total	Level------- Metric Name
0.00	0	45	4-------- MACHARITHINDEP
0.00	0	4	5------- PACKAGE_ARITH_INDEP
		1	6------ NO_MAX_INT
		1	6------ NO_MIN_INT
		0	6------ NO_MAX_DIGITS
		0	6------ NO_MAX_MANTISSA
		0	6------ NO_FINE_DELTA
		2	6------ NO_TICK
		0	5------- NO_INTEGER_DECL
		0	5------- NO_SHORT_INTEGER_DECL
		0	5------- NO_LONG_INTEGER_DECL
		0	5------- NO_FLOAT_DECL
		0	5------- NO_SHORT_FLOAT_DECL
		0	5------- NO_LONG_FLOAT_DECL
		0	5------- NO_NATURAL_DECL
		41	5------- NO_POSITIVE_DECL
	0	0	5------- FIXED_CLAUSE
0.83	358	432	4-------- MACHREPINDEP
		1	5------- NO_PRAGMA_PACK
0.14	10	74	5------- NUMERIC_CONSTANT_DECL
1.00	3	3	5------- NUMERIC_TYPE_DECLARATIONS
0.97	345	354	5------- CLAUSE_REP_INDEP
0.99	201	204	6------ NO_LENGTH_CLAUSE_FOR_SIZE
1.00	4	4	6------ NO_LENGTH_CLAUSE_FOR_STORAGE_SIZE
0.96	70	73	6------ NO_ALIGNMENT_CLAUSE_FOR_RECORD_TYPES
0.96	70	73	6------ NO_COMPONENT_CLAUSE_FOR_RECORD_TYPES
0.98	17296	17573	4-------- MACHCONFIGINDEP
0.99	1103	1109	5------- NO_ADDRESS_CLAUSE_IN_DECL
		0	5------- NO_PRAG_SYS_PARAM
0.98	16193	16464	5------- NO_REP_ATTRIBUTE
1.00	4552	4552	4-------- MACHCODEINDEP
1.00	4552	4552	5------- NO_MACH_CODE_STMT
0.99	16667	16683	3--------- SOFT_INDEP
1.00	16464	16464	4-------- NO_SYS_DEP_MOD
		3	4-------- NO_IMPL_DEP_PRAGMAS
		12	4-------- NO_PRAGMA_INTERFACE
0.99	203	204	4-------- NON_ACCESS_TYPE
		0	4-------- NO_IMPL_DEP_ATTRS
0.00	0	4	3--------- PHYS_LIM_INDEP
		4	4-------- COMPILER_LIMIT
0.68	1056	1548	2---------- MODULARITY
0.62	765	1226	3--------- INFORMATION_HIDING
0.71	757	1073	4-------- HIDDEN_INFORMATION
0.68	93	136	5------- CONSTANTS_HID
	0	0	5------- EXCEPTIONS_HID
0.92	595	646	5------- VARIABLES_HID
0.28	58	204	5------- TYPES_HID
0.13	11	84	5------- SUBTYPES_HID
0.00	0	3	5------- TASKS_HID
0.05	8	153	4-------- PRIVATE_INFORMATION
0.01	2	146	5------- PRIVATE_TYPES
0.50	1	2	5------- LIMITED_PRIVATE_TYPES
1.00	2	2	5------- PRIVATE_TYPE_AND_PART
1.00	3	3	5------- PRIVATE_TYPE_AND_CONSTANT
0.97	253	260	3--------- PROFILE
0.97	253	260	4-------- LIMITED_SIZE_PROFILE

Date: 12-30-1991 page: 3
DUA0:[USER.ADAMAT.UH1_DATA]GAT_AND_SERVICES.REP_COM;1

Score	Good	Total	Level------- Metric Name
	0	0	4-------- SIMPLE_BLOCKS
0.61	38	62	3--------- COUPLING
0.58	18	31	4-------- NO_MULTIPLE_TYPE_DECLARATIONS
0.65	20	31	4-------- NO_VARIABLE_DECLARATIONS_IN_SPEC
0.88	6810	7774	2---------- SELF_DESCRIPTIVENESS
0.81	3560	4386	3--------- COMMENTS
0.91	2484	2722	4-------- N_COMMENTS
0.99	744	746	5------- NCS_SPEC
1.00	155	155	6------ NCS_PACKAGE_SPEC
0.87	13	15	6------ NCS_TASK_SPEC
1.00	576	576	6------ NCS_SUBPROG_SPEC
0.98	709	727	5------- NCS_BODY
1.00	100	100	6------ NCS_PACKAGE_BODY
1.00	15	15	6------ NCS_TASK_BODY
1.00	594	594	6------ NCS_SUBPROG_BODY
0.00	0	15	6------ NCS_SUBUNIT
0.00	0	3	6------ NCS_BODY_STUB
0.52	208	397	5------- NCS_STATEMENTS
0.00	0	20	6------ NCS_EXIT
0.01	1	148	6------ NCS_RETURN
	0	0	6------ NCS_GOTO
	0	0	6------ NCS_ABORT
1.00	1	1	6------ NCS_DELAY
	0	0	6------ NCS_TERMINATE
0.84	118	140	6------ NCS_WITH
1.00	88	88	6------ NCS_USE
0.97	823	852	5------- NCS_DECLARATIONS
1.00	21	21	6------ NCS_PRAGMA
0.22	2	9	6------ NCS_RECORD_REPRESENTATION
0.89	16	18	6------ NCS_ADDRESS_CLAUSE
0.00	0	9	6------ NCS_ALIGNMENT_CLAUSE
0.44	4	9	6------ NCS_LENGTH_CLAUSE
1.00	134	134	6------ NCS_CONSTANT_DECL
1.00	640	640	6------ NCS_VARIABLE_DECL
0.50	6	12	6------ NCS_ENTRY_DECL
0.65	1076	1664	4-------- N_COMMENTED
0.71	160	226	5------- NCO_SPEC
1.00	31	31	6------ NCO_PACKAGE_SPEC
1.00	3	3	6------ NCO_TASK_SPEC
0.66	126	192	6------ NCO_SUBPROG_SPEC
0.96	215	225	5------- NCO_BODY
1.00	20	20	6------ NCO_PACKAGE_BODY
1.00	3	3	6------ NCO_TASK_BODY
0.97	192	198	6------ NCO_SUBPROG_BODY
0.00	0	3	6------ NCO_SUBUNIT
0.00	0	1	6------ NCO_BODY_STUB
0.14	54	397	5------- NCO_STATEMENTS
0.00	0	20	6------ NCO_EXIT
0.01	1	148	6------ NCO_RETURN
	0	0	6------ NCO_GOTO
	0	0	6------ NCO_ABORT
1.00	1	1	6------ NCO_DELAY
	0	0	6------ NCO_TERMINATE
0.26	37	140	6------ NCO_WITH
0.17	15	88	6------ NCO_USE
0.79	647	816	5------- NCO_DECLARATIONS

Date: 12-30-1991 page: 4
DUA0:[USER.ADAMAT.UH1_DATA]GAT_AND_SERVICES.REP_COM;1

Score	Good	Total	Level------- Metric Name
0.19	4	21	6------ NCO_PRAGMA
0.67	2	3	6------ NCO_RECORD_REPRESENTATION
0.67	4	6	6------ NCO_ADDRESS_CLAUSE
0.00	0	3	6------ NCO_ALIGNMENT_CLAUSE
0.67	2	3	6------ NCO_LENGTH_CLAUSE
0.96	128	134	6------ NCO_CONSTANT_DECL
0.78	501	640	6------ NCO_VARIABLE_DECL
1.00	6	6	6------ NCO_ENTRY_DECL
0.96	3250	3388	3--------- IDENTIFIER
0.96	3250	3388	4-------- NO_PREDEFINED_WORDS
0.54	6580	12179	2---------- SIMPLICITY
0.87	571	657	3--------- CODING_SIMPLICITY
0.99	571	579	4-------- SIMPLE_BOOLEAN_EXPRESSION
0.00	0	78	4-------- EXPRES_TO_DO_BOOLEAN_ASSIGN
0.27	1053	3846	3--------- DESIGN_SIMPLICITY
0.48	412	855	4-------- CALLS_TO_PROCEDURES
0.93	39	42	4-------- ARRAY_TYPE_EXPLICIT
0.56	84	151	4-------- SUBTYPE_EXPLICIT
0.52	17	33	4-------- ARRAY_RANGE_TYPE_EXPLICIT
0.18	501	2765	4-------- DECLARATIONS_CONTAIN_LITERALS
0.65	4956	7676	3--------- FLOW_SIMPLICITY
0.54	566	1040	4-------- BRANCH_CONSTRUCTS
0.84	167	198	4-------- SINGLE_EXIT_SUBPROGRAM
0.51	47	93	4-------- FOR_LOOPS
0.98	255	260	4-------- LEVEL_OF_NESTING_BY_MODULE
0.90	311	345	4-------- LEVEL_OF_NESTING
0.66	686	1040	4-------- STRUCTURED_BRANCH_CONSTRUCT
0.91	947	1040	4-------- NON_BACK_BRANCH_CONSTRUCT
	0	0	4-------- NO_LABELS
0.68	490	718	4-------- DECISIONS
	0	0	4-------- GOTOS
0.43	1040	2422	4-------- BRANCH_AND_NESTING
0.87	225	260	4-------- CYCLOMATIC_COMPLEXITY
0.85	222	260	4-------- MULTIPLE_COND_CYCLOMATIC_COMPLEXITY
0.78	3866	4952	2---------- SYSTEM_CLARITY
0.78	3866	4952	3--------- STYLE
0.74	957	1295	4-------- EXPRESSION_STYLE
0.99	571	579	5------- NON_NEGATED_BOOLEAN_EXPRESSIONS
0.54	312	576	5------- EXPRESSIONS_PARENTHESIZED
0.69	64	93	5------- NO_WHILE_LOOPS
0.21	10	47	5------- FOR_LOOPS_WITH_TYPE
0.97	2204	2264	4-------- DECLARATION_STYLE
0.90	456	509	5------- NO_DEFAULT_MODE_PARAMETERS
	0	0	5------- PRIVATE_ACCESS_TYPES
0.99	1745	1752	5------- SINGLE_OBJECT_DECLARATION_LISTS
1.00	3	3	5------- SINGLE_IMPLICIT_TYPE_ARRAY
	0	0	5------- NO_INITIALIZATION_BY_NEW
0.57	288	501	4-------- NAMING_STYLE
0.00	0	93	5------- STRUCTURES_NAMED
0.00	0	93	6------ NAMED_LOOPS
	0	0	6------ NAMED_BLOCKS
0.72	254	351	5------- STRUCTURE_ENDS_WITH_NAME
0.98	254	258	6------ MODULE_END_WITH_NAME
0.00	0	93	6------ LOOP_END_WITH_NAME
	0	0	6------ BLOCK_END_WITH_NAME
0.00	0	20	5------- NAMED_EXITS

Date: 12-30-1991 page: 5
DUA0:[USER.ADAMAT.UH1_DATA]GAT_AND_SERVICES.REP_COM;1

ADAMAT Results

Score	Good	Total	Level------- Metric Name
0.92	34	37	5------- NAMED_AGGREGATE
0.47	417	892	4-------- QUALIFICATION_STYLE
0.00	0	37	5------- QUALIFIED_AGGREGATE
0.49	417	855	5------- QUALIFIED_SUBPROGRAM

REFERENCES

1. Jon D. Valett and Frank E. McGarry, "A Summary of Software Measurement Experiences in the Software Engineering Laboratory", The Journal of System and Software, Vol. 9, pages 137-148, 1989.

2. Frank McGarry, Linda Esker, Kelvin Quimby, "Evolution of Ada Technology in a Production Environment", Goddard Space Flight Center, Greenbelt, MD, 1988.

3. Frank McGarry, Sharon Waligora, "Experiments in SE Technology: Recent Studies in the SEL", Software Engineering Laboratory, December 1991.

4. IIT Research Institute, Test Case Study: Estimating the Cost of Ada Software Development, April 1989.

5. Kenneth J. Lee, et. al., An OOD Paradigm for Flight Simulators, 2nd Edition, CMU/SEI-88-TR-30, September 1988.

6. IIT Research Institute, A Descriptive Evaluation of Software Sizing Models, September 1987.

7. Boeing Aerospace Company, Specification of Software Quality Attributes Software Quality Evaluation Handbook, RADC-TR-85-37, Volume III (of three), February 1985.

8. J. D. Anderson, J. A. Perkins, "Experience Using an Automated Metrics Framework in the Review of Ada Source for WIS", Proceeding of the 6th National Conference on Ada Technology, 1988.

9. Ken Zwanzig, ed., Handbook for Estimating Using Function Points, GUIDE International, November, 1984.